uc3m | Universidad Carlos III de Madrid

Grado en Ingeniería en Tecnologías de Telecomunicación
2018-2019

Trabajo Fin de Grado

"Diseño y Desarrollo de una Unidad Terminal Remota en FPGA"

Alicia Bermejo Rodríguez

Tutor/es

Marta Portela García

Leganés, 3 de Julio de 2019

SENER

RESUMEN

El proyecto desarrollado se basa en el diseño y la implementación de una Unidad Terminal Remota (RTU) para el control térmico. La propuesta de este proyecto es la utilización de esta unidad en la implementación de un nuevo satélite de bajo coste realizado por la Universidad Carlos III y la empresa española SENER.

Para la realización de la función del control de temperatura se ha utilizado una FPGA y un software específico. También se ha usado para el diseño, síntesis y el análisis de los circuitos digitales. Se realiza en lenguaje de descripción hardware VHDL.

Primero se realiza el diseño de cada uno de los bloques y componentes que forman parte de la RTU en VHDL. Posteriormente, se realizan comprobaciones de diversas formas (simulación, matemáticamente, montaje de un sistema…). Todas estas pruebas se hacen para verificar el funcionamiento de cada una las partes que componen el proyecto.

Palabras clave: *Remote Terminal Unit*, satélites, sensores térmicos, FPGA, termistores

ABSTRACT

The project is based on the design and implementation of a Remote Terminal Unit (RTU) for thermal control. The proposal of this project is the use of this unit in the implementation of a new low cost satellite made by the Carlos III University and the Spanish company SENER.

A FPGA and a specific software have been used for the design, synthesis and analysis of digital circuits. Also, they have been used for the temperature control function. It is done in VHDL hardware description language.

Firstly, it is made the design of each of the blocks and components that are part of the RTU in VHDL. Subsequently, verifications are checked in several ways (simulation, mathematically, assembly of an extern system ...). All of these tests are made to verify the operation of each of the parts that make up the project.

Keywords: Remote Terminal Unit, satellite, thermal sensors, FPGA, thermistors.

AGRADECIMIENTOS

Deseo expresar mi agradecimiento a todos los que me han acompañado y ayudado durante el tiempo que ha durado la carrera y la elaboración de este trabajo.

En especial me gustaría dar las gracias a mi tutora Marta Portela García, por haber estado siempre disponible para ayudarme, aconsejarme y orientarme en el desarrollo del trabajo; y por haberme ofrecido la oportunidad de formar parte de un proyecto tan amplio.

A mis padres, a mi hermana, a mi familia y amigos por apoyarme durante esta etapa académica, por desearme siempre lo mejor y por estar ahí en cualquier momento.

A la Universidad Carlos III por ofrecer a los estudiantes la posibilidad de colaborar en proyectos con empresas, como es la Cátedra de investigación UC3M-SENER.

Gracias a todos.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

LISTA DE ABREVIATURAS

ADC	*Analog to Digital Converter* (Conversor anlógico digital)
AOCS	*Attitude and Orbit Control System*
CR	Retorno de Carro
ECSS	*European Cooperation for Space Standardization* (Cooperación Europea para la Normalización Espacial)
FPGA	*Field Programmable Gate Array*
GND	*Ground*
HDL	*Hardware Description Language*
IEEE	*Institute of Electrical andElectronics Engineers*
LF	Fin de línea
NASA	*National Aeronautics and Space Administration* (Administración Nacional de la Aeronáutica y del Espacio)
OBC	*On Board Computer*
OTAN	Organización del Tratado del Atlántico Norte
PWM	*Pulse-Width Modulation*
RTD	*Resistance Temperature Detector*
RTU	*Remote Terminal Unit* (Unidad Terminal Remota)
UART	*Universal Asynchronous Receiver and Transmitter*
UC3M	Universidad Carlos III de Madrid
UNOOSA	Oficina de Naciones Unidas para Asuntos del Espacio Exterior
VHDL	Acrónimo combinación de VHSIC y HDL
VHSIC	*Very High Speed Integrated Circuit*

1 INTRODUCCIÓN

1.1 Motivación.

Alrededor del planeta Tierra hay numerosos satélites artificiales girando. Actualmente, se encuentran aproximadamente 4.921 satélites orbitando la Tierra según la Oficina de Naciones Unidas para Asuntos del Espacio Exterior (UNOOSA). Todos estos satélites situados en el espacio no están en funcionamiento, aunque continúan en órbita. Cuando la vida útil del satélite se ha terminado, se convierten en basura espacial.[1] [2]

Existen diferentes tipos de satélites artificiales según la misión realizada por cada uno de ellos. La clasificación es: [3] [4]

- Satélites Científicos: Tienen diversos objetivos científicos como el estudio de la magnetosfera terrestre (satélites MMS de la NASA) o analizar el fondo marino de los océanos (satélite Seasat de la NASA).
- Satélites de Comunicaciones: Es una aplicación muy difundida en la actualidad. Se utilizan para la transmisión de radio, televisión, telefonía e Internet en todo el mundo, mediante la emisión de señales. Un ejemplo de este tipo es el satélite Hispasat 1A.
- Satélites de Meteorología: Se utilizan para la observación de la atmósfera terrestre. Un ejemplo es el satélite Meteosat.
- Satélites de Navegación: Se encargan de la identificación de diferentes localizaciones terrestres. Un ejemplo es el sistema GPS, que tiene tres satélites.
- Satélites de teledetección: Se utilizan para la observación del planeta para estudios tanto terrestres como marítimos. Un ejemplo de este tipo de satélites es el Landsat 8.
- Satélites Militares y de espionaje: Se utilizan en ciertos países para las operaciones militares pero con misiones secretas. Un ejemplo de este tipo es el satélite español PAZ.

Hoy en día se está desarrollando una nueva filosofía conocida como “New Space” o “Space Industry 4.0”. Se crean satélites con menor coste de producción, en un período de tiempo reducido y utilizando modelos más flexibles, gracias al desarrollo de nuevas tecnologías. [5] Reducir el peso del satélite es fundamental

para reducir el coste puesto que el precio del lanzamiento depende de su peso. Los nanosatélites, son satélites de bajo coste que pueden realizar distintos tipos de misiones, fundamentalmente científicas. Hay un desarrollo creciente de satélites de este tipo debido a su fácil acceso para empresas de cualquier tamaño y para investigaciones universitarias. [6]–[11]

Los satélites convencionales hasta ahora tendían a ser cada vez más grandes y con una tecnología más sofisticada. Se tardaba entre 5 y 15 años en desarrollarlos, y tenían unos costes de producción muy elevados que podían llegar a los 500 millones de euros. Estos satélites tradicionales tienen unas dimensiones considerables, para que realicen más funciones. Además, como tienen que tener una durabilidad mínima de 20 años, y sus elementos no pueden ser reemplazados una vez lanzados, entonces el coste para garantizar la confianza en ellos se eleva. [6]

Fig. 1 Satélite artificial convencional [1]

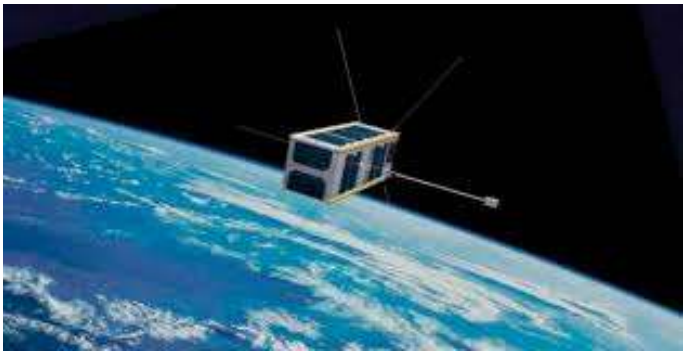


Fig. 2 Nanosatélite [12]

Este trabajo de fin de grado se ha desarrollado en el marco de la Cátedra de investigación que tiene la Universidad Carlos III de Madrid con la empresa española SENER.

La empresa SENER es un referente a nivel internacional en el sector aeroespacial. Dentro de esta Cátedra hay aproximadamente treinta trabajos de fin de grado y fin de máster, entre los que se encuentra el presente Trabajo de Fin de Grado. El objetivo final de esta Cátedra es el diseño, la construcción y el lanzamiento de un satélite *low-cost* por parte de la UC3M y SENER.

El satélite diseñado en la Cátedra UC3M-SENER será un satélite de bajo coste científico en el que se probarán nuevas tecnologías para el espacio. El tiempo de diseño y construcción estimado para este satélite es mucho menor que el de los satélites convencionales. Se quiere intentar llevar a cabo el lanzamiento en el año 2022.

El diagrama de bloques que se muestra a continuación contiene todos los bloques y componentes que forman parte del satélite y que forman parte de las actividades a desarrollar en la Cátedra, en el lado de carga (*Payload side*), en el lado de la plataforma (*Platform side*) y en el segmento de tierra (*Ground Segment*), así se tiene una visión general de la situación de la *Remote Terminal Unit* (RTU) en la parte central del lado de la plataforma. Tiene conexiones con los *heaters*, los sensores solares, los termistores, los magneto torquers y el ordenador de a bordo.[13]

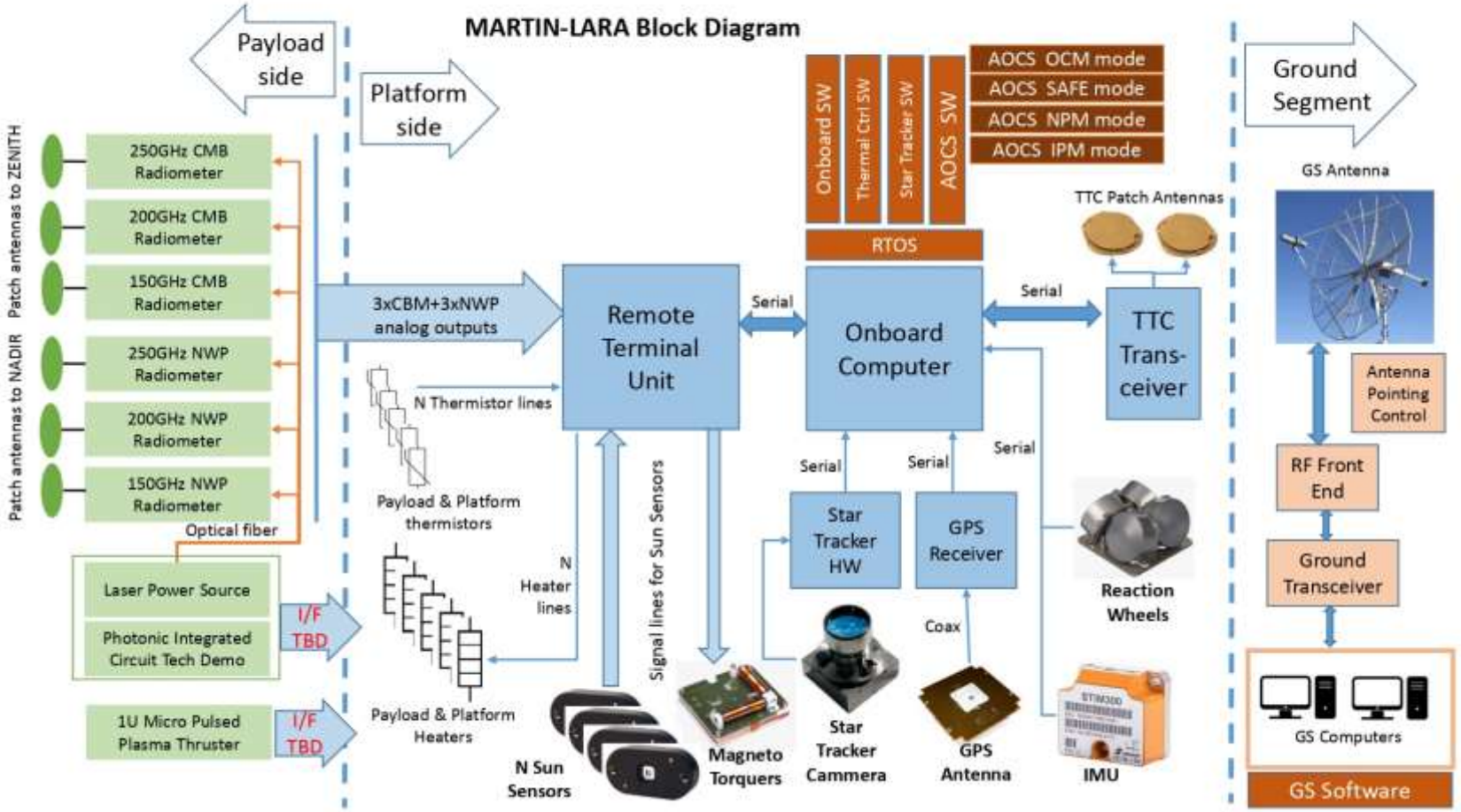


Fig. 3 Diagrama de bloques de un satélite [13]

1.2 Objetivos.

Este proyecto se centra en el diseño y desarrollo de una Unidad Terminal Remota o *Remote Terminal Unit* (RTU) en *Field Programmable Gate Array* (FPGA).

La RTU es una unidad presente en las naves espaciales y satélites de tamaño medio y grande. Se encarga principalmente de interconectar sensores, actuadores e interfaces digitales con el ordenador de a bordo (*On Board Computer,* OBC). Además, efectúa el control de los actuadores y la adquisición de los datos analógicos, es decir, es un sistema de control distribuido en el satélite. Finalmente, es una unidad conectada al OBC mediante comunicación serie y a la vez está controlada por él. [14]–[17]

En este trabajo, la tarea que debe realizar la RTU es el control de temperatura. Las capacidades que tiene la RTU y que se deben implementar para conseguir el objetivo principal son las siguientes:[18]

- El sistema se encarga de la adquisición de la temperatura con sensores térmicos.
- Se encarga del procesamiento de los datos adquiridos para tomar decisiones.
- Se implementa el algoritmo de control.
- Se encarga del control de los actuadores o *heaters* para poder cerrar el lazo y controlar.
- Tiene un sistema de comunicación para comunicarse con el OBC y tiene una lógica de control específica que le permite realizar funciones específicas para tomar decisiones de forma independiente. Sirve para configurar las características del sistema y para enviar los datos de las medidas realizadas.

1.3 Entorno de Desarrollo.

Desde el principio, se ha decidido utilizar una *Field Programmable Gate Array* (FPGA) para realizar el diseño e implementación de la RTU, en vez de utilizar un microcontrolador.

Las FPGAs son dispositivos programables para describir circuitos digítateles utilizando un lenguaje específico, VHDL o Verilog. Contienen bloques de lógica que se programan para desarrollar una funcionalidad determinada y trabajan con procesos en paralelo. [19]–[21]

Los microcontroladores son procesadores con circuitos integrados que se pueden programar. Trabajan con procesos secuenciales. [19]

La razón por la que se eligió la FPGA es para poder realizar muchas tareas en paralelo, mientras que con un microcontrolador los procesos se ejecutan de manera secuencial, lo que implica que hasta que no termina uno no comienza el siguiente. Así, posteriormente se podrán añadir nuevas funcionalidades sin problemas. [19]

Para programar y configurar la FPGA se ha utilizado el Lenguaje de Descripción Hardware VHDL.

1.4 Estructura del documento.

Este trabajo de fin de grado está organizado en nueve capítulos en los cuales se explican diferentes aspectos del proyecto realizado.

El primer capítulo es la introducción, en él se expone la motivación para realizar este trabajo, los objetivos que se tienen que llevar a cabo y el entorno de desarrollo que se utiliza en el proyecto.

El segundo capítulo es el marco regulador aplicado al desarrollo del proyecto.

El tercer capítulo es el estado del arte. Se analizan los componentes utilizados en el trabajo, su descripción teórica y la función del control térmico en otros satélites.

El cuarto capítulo son las especificaciones, en él se exponen las especificaciones técnicas proporcionadas que se deben cumplir en el diseño e implementación de la RTU.

El quinto capítulo es el diseño realizado, en él se exponen los diagramas de bloques, máquinas de estados y decisiones de diseño tomadas para el funcionamiento de la RTU.

El sexto capítulo son las validaciones y pruebas experimentales, en él se explican en detalle cada una de las pruebas realizadas para comprobar el correcto de funcionamiento de las diferentes partes del sistema.

El séptimo capítulo son las conclusiones y trabajos futuros, en él se explican las conclusiones obtenidas del desarrollo de este trabajo así como posibles trabajos futuros con los que se podría continuar.

El octavo capítulo es el presupuesto necesario para el desarrollo del proyecto.

El noveno capítulo es la bibliografía, incluye las referencias utilizadas para la elaboración del trabajo.

2 MARCO REGULADOR

La propuesta del trabajo es el diseño y desarrollo de una RTU en FPGA para satélites. Las normas implicadas en la realización del proyecto incluyen por un lado las normas del lenguaje de programación utilizado y por otro las certificaciones del sector espacial.

El lenguaje de descripción hardware utilizado, VHDL, sigue un estándar del *Institute of Electrical and Electronics Engineers* (IEEE). El estándar aprobado actualmente es el *IEEE Standard VHDL Language Reference Manual* de 2008 (IEEE Std 1076-2008), aunque existe un borrador posterior de junio de 2018, aún sin aprobar. [22]–[24]

En el estándar anterior se recoge toda la información y normas sobre la utilización de este lenguaje para la creación de sistemas electrónicos.

En el sector aeroespacial existen numerosas normas y certificaciones. El desarrollo del satélite se realiza en Europa entonces la normativa vigente la marca la Agencia Espacial Europea (ESA).

Desde el año 1993, las diferentes agencias e industrias espaciales europeas acordaron la unificación de los estándares, para ello comenzaron la Cooperación Europea para la Normalización Espacial (ECSS).[25] Se incluyen numerosos estándares de certificación tanto para los materiales que componen el satélite, los programas y funciones del mismo. Se trabajan en función de disciplinas porque son normas que se van revisando y actualizando con el tiempo. Por ejemplo el ECSS-E-ST-10-06C- incluye la especificación de requisitos técnicos o el ECSS-E-ST-10-09C es sobre el Sistema de coordenadas de referencia. [26]

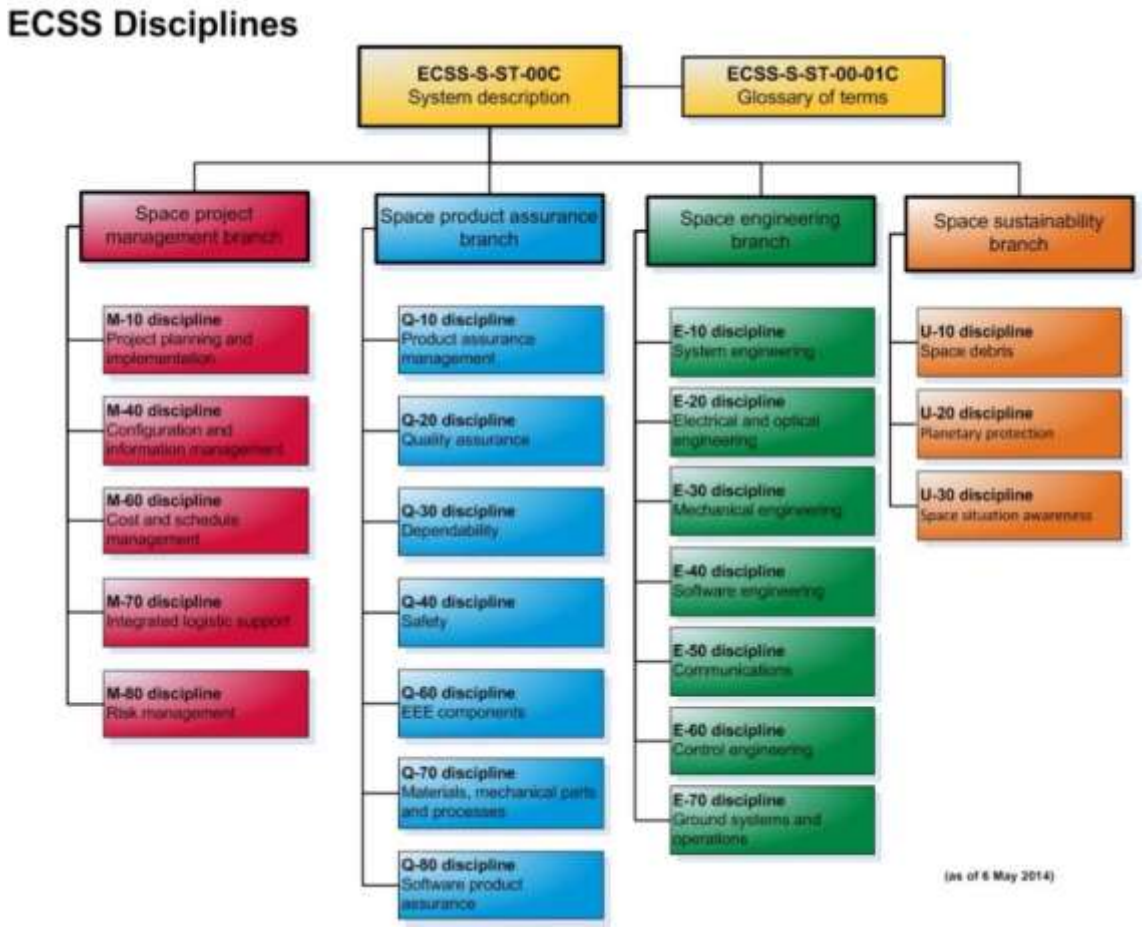


Fig. 4 Disciplinas ECSS [27]

La regulación está basada en el ámbito (civil o militar) y en la misión que va a desarrollar el satélite. Además de la regulación europea, se podrían aplicar otro tipo de regulaciones, por ejemplo si fuera de uso militar la Organización del Tratado del Atlántico Norte (OTAN) podría aplicar también otras regularizaciones y estandarizaciones. [27]

En el desarrollo de este trabajo no se han tenido en cuenta estos estándares y normalizaciones porque se ha realizado únicamente el diseño de la RTU de manera teórica. Cuando se realice la implementación de la misma y la integración de todos los componentes del satélite se velaría por la aplicación de estos estándares. Entonces, estas regularizaciones están fuera del alcance del trabajo y no se tienen que considerar.

3 ESTADO DEL ARTE

3.1 Descripción de componentes.

3.1.1 Sensores de temperatura.

Los sensores de temperatura son dispositivos con la capacidad de detectar cambios en la temperatura del medio, estas variaciones son transformadas en señales eléctricas procesadas por un dispositivo electrónico. [28]

Este tipo de sensores tienen principalmente tres partes: un elemento sensor, una vaina de material conductor en su interior y un cable para conectarlo al sistema electrónico. [28]

Uno de los tipos de sensores de temperatura son los sensores de temperatura resistivos, formado por los termistores y los dispositivos RTD (*Resistance Temperature Detector*).[29]

Fig. 5 Termistor [30]

Los RTD son sensores basados en conductores, los más frecuentes son los que tienen una resistencia de platino. Su principal ventaja es la linealidad en un amplio rango de temperaturas. [29]

Los termistores son semiconductores electrónicos con un coeficiente de temperatura de resistencia positivo o negativo. La resistencia de los semiconductores se modifica con la temperatura, es decir, el número de portadores varía. La relación entre la temperatura, la resistencia y el número de portadores es

la siguiente, si aumenta la temperatura, aumenta el número de portadores y disminuye la resistencia. [29]

Para obtener la respuesta del sensor de temperatura en función de la temperatura se utiliza la ecuación de Steinhart-Hart, aunque no es tan exacto como los datos de los fabricantes proporcionados en los *datasheets* es una aproximación buena: [22]

$$\frac{1}{T} = \frac{1}{To} + \frac{1}{\beta} \cdot \ln\left(\frac{Rth}{Ro}\right) \rightarrow \beta \cdot \left(\frac{1}{T} - \frac{1}{To}\right) = \ln\frac{Rth}{Ro} \rightarrow Rth = Ro \cdot e^{\left(\beta \cdot \left(\frac{1}{T} - \frac{1}{To}\right)\right)} \tag{3.1}$$

donde:

T: Temperatura absoluta

To: Temperatura absoluta de referencia

β: Constante de temperatura

Rth: Resistencia del termistor a la temperatura absoluta

Ro: Resistencia a la temperatura de referencia

Para comprobar la linealidad de la resistencia del sensor con respecto a la temperatura se utiliza el parámetro β más simple que es 3380, Ro de 10KΩ y un valor de To de 25°C (298.15°K). Se hace en el rango de temperaturas especificado, entre -10°C y +50°C.

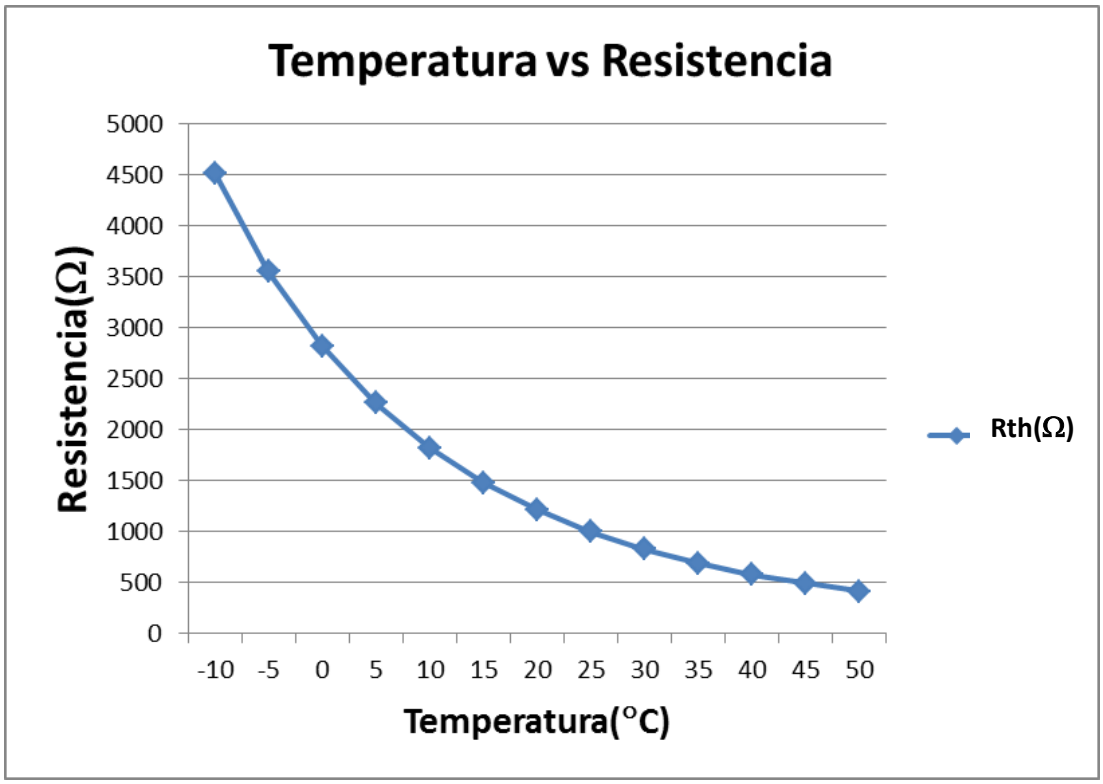


Fig. 6 Gráfica de la resistencia del termistor en función de la temperatura.

Se utiliza un conversor analógico digital para obtener los datos porque no se pueden obtener los valores de la resistencia del sensor de temperatura directamente, se emplea un circuito divisor de tensión para obtenerlos.

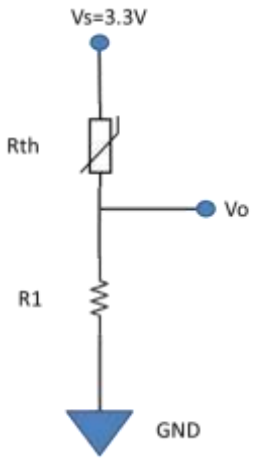


Fig. 7 Circuito divisor de tensión.

Con el sistema electrónico que recibe las variaciones de temperatura detectadas en el sensor no se mide el valor de temperatura directamente, sino la tensión asociada a esa temperatura.

La tensión medida a la salida se calculará mediante la fórmula [22]:

$$Vo = Vs \cdot \frac{R1}{R1 + Rth} \tag{3.2}$$

Una vez obtenido el valor de la resistencia del sensor en función de la temperatura, se obtiene la curva de temperatura- tensión linealizada. La tensión de la fuente es de 3.3V, que es la tensión máxima de la FPGA, la R1 elegida es de 10KΩ y la resistencia interna del termistor a 25°C es de 10KΩ.

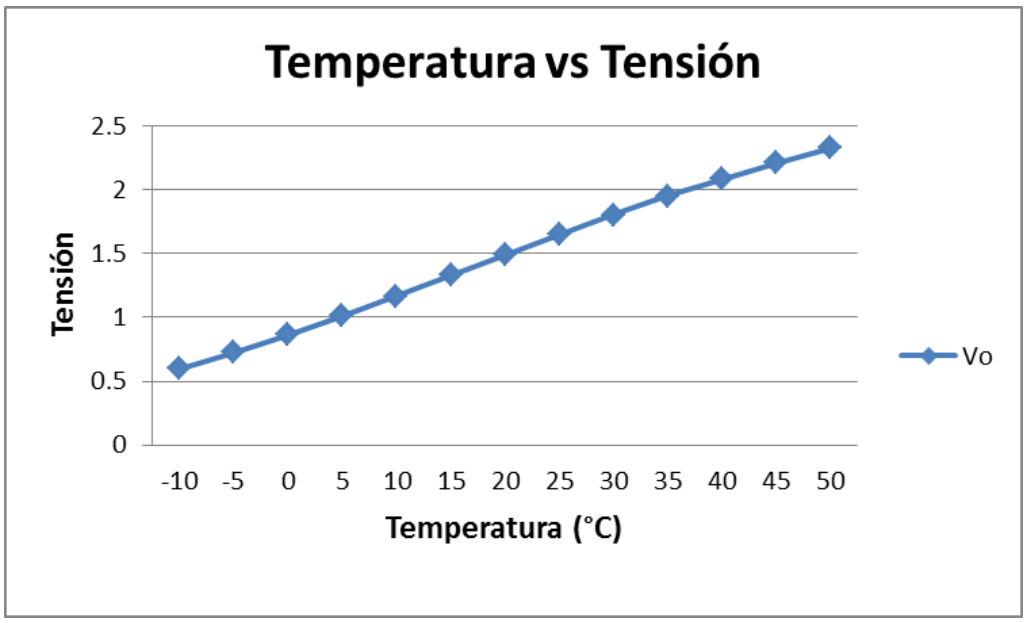


Fig. 8 Gráfica de la tensión medida en función de la temperatura.

La linealidad está claramente marcada para los valores analizados. Se utilizan para mediciones de temperatura de alta precisión por su sensibilidad a los cambios de temperatura.

3.1.2 Actuadores (*heaters*)

Los actuadores o *heaters* se suelen utilizar en el control de la temperatura del satélite. Estos dispositivos mantienen la temperatura de las baterías o cargas útiles de los satélites en unos valores adecuados en los períodos de frío de la órbita, como pueden ser los eclipses.

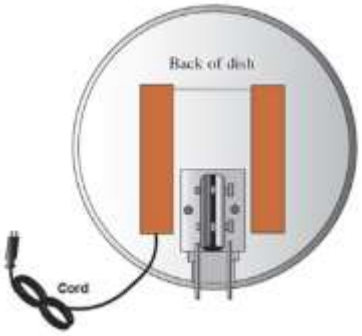


Fig. 9 Heater [31]

3.1.3 Placas de implementación.

Para el desarrollo del proyecto se podrían utilizar diversas placas para realizar la implementación. Se eligió utilizar una FPGA. [19]–[21]

La elección de la FPGA fue realizada a partir de las ventajas que presenta respecto a un microcontrolador.

- La FPGA puede realizar un mayor número de operaciones en cada ciclo de reloj que un microcontrolador.
- La FPGA realiza operaciones en paralelo, se pueden realizar más actividades simultáneamente por tanto la velocidad de procesamiento de datos es mayor. Los microcontroladores actúan de manera secuencial, solo realizan una tarea al mismo tiempo.
- La FPGA se puede reprogramar y modificar los circuitos lógicos para cambiar las funciones que realiza, en cambio los microcontroladores tienen una circuitería interna estática.

3.2 Control Térmico.

El control de temperatura del satélite es una función esencial. La temperatura del satélite debe tener determinado rango para que los componentes que lo constituyen (equipos electrónicos, carga útil…) funcionen correctamente y de manera óptima. El sistema de control térmico se encarga de mantener una temperatura adecuada en el interior del satélite. [32]–[35]

El control térmico se puede realizar de dos formas mediante sistemas pasivos y mediante sistemas activos.

Los sistemas pasivos no necesitan potencia de entrada para controlar la temperatura interna del satélite. Se utilizan en nanosatélites por su bajo coste, volumen, peso y riesgo. Algunos sistemas pasivos son los recubrimientos térmicos o las tuberías térmicas.

Los sistemas activos tienen mayor precisión y fiabilidad, depende de la potencia de entrada. En este tipo de sistemas se incluyen los *heaters* y los refrigeradores entre otros. Se utilizan en satélites de gran tamaño y en limitadas ocasiones en los nanosatélites, para los componentes que necesitan un rango de temperaturas determinado.

3.3 *Remote Terminal Units* (RTU)

Las RTUs son unidades que están presentes en los satélites de tamaño medio y grande. Descargan los datos de los actuadores y los termistores en el ordenador de a bordo (OCB). [14], [15], [36]

Dependiendo del satélite pueden tener varias RTUs como el satélite SMART-1 de la ESA, que cuenta con cuatro unidades terminales remotas. Cada una de ellas tiene su propio OCB con el que se comunica a través de un bus de datos. [37]

La ESA tiene una visión del OBC y la RTU como un sistema centralizado. Apoya un sistema integrado por el sistema de control, la memoria, el microprocesador y todas las interfaces, no únicamente con el bus de conexión con el OBC. Este diseño incluye el OBC y la carga útil.[38]

4 ESPECIFICACIONES

El diseño y desarrollo de la unidad terminal remota con capacidad de procesamiento de datos tiene algunas especificaciones técnicas, que deben cumplirse. Esta unidad debe realizar la función de control térmico del satélite.

El diseño tiene varios bloques con distintas entradas, salidas y funcionalidades. A continuación, se desarrolla cada uno de los bloques con sus especificaciones propias, para posteriormente detallar el diseño realizado de la RTU y su implementación en lenguaje hardware VHDL. [18]

4.1 Adquisición de temperatura.

El bloque de adquisición de temperatura se encarga de adquirir los datos de temperatura mediante tres sensores de temperatura. La adquisición de los datos se realiza periódicamente, cada segundo.

Los sensores de temperatura que se utilizarán para medir la temperatura serán de tipo PT1000. Son dispositivos RTD construidos con una resistencia de platino, la resistencia del sensor varía con la temperatura de forma lineal, si aumenta la temperatura aumenta la resistencia. Se utilizan estos dispositivos por su sensibilidad, precisión y fiabilidad. El rango de temperatura del sensor de platino es de -200°C a 850°C, mayor que el de otros materiales utilizados para los sensores RTD. [29] La resistencia de estos sensores es de 1000Ω a 0°C.

En la siguiente tabla se muestran los materiales usados para los RTD, con sus características de temperatura y precisión: [29]

Tabla 1 Materiales de los Resistance Temperature Detector (RTD) [29]

Material	**Rango Temperatura (°C)**	**Variación (%/°C) A 25°C**
Platino	-200 a 850	0.39
Níquel	-80 a 320	0.67
Cobre	-200 a 260	0.38
Níquel- acero	-200 a 260	0.46

Los datos obtenidos de los termistores se deben almacenar en registros para poder procesarlos y utilizarlos después para la detección de fallos en el sistema y para disparar una alarma.

4.2 Gestión de tiempo.

La gestión del tiempo se utiliza para la sincronización del sistema completo. Además, todos los datos guardados en los diferentes registros se han identificado con una etiqueta de tiempo, que indica el momento en el que se guardaron cada uno de los valores.

4.3 Interfaz de comunicación con el usuario.

Es necesaria una interfaz de comunicación con el ordenador, tanto para transmitir como para recibir datos.

Por un lado, se escribieron los valores de configuración del sistema. Los parámetros a configurar fuero: T_{min}, T_{max}, T_{target}, P_{heater}, C_1, C_2, T_{ctrl}, P_{limit}, $Tiempo_0$ y T_{meas_def}. Estos parámetros de configuración se utilizaron para realizar comprobaciones y activar alarmas en caso de fallos.

Por otro lado, a través de la interfaz es posible leer los resultados obtenidos. Se almacenaron los datos en diferentes registros y posteriormente leyeron los valores de los mismos.

4.4 Algoritmo de Control.

La primera tarea realizada es la verificación de las temperaturas. Para ello, se realiza una votación por mayoría de las temperaturas adquiridas por los tres termistores. Tras la ejecución del algoritmo se obtiene el valor del ciclo de trabajo de salida.

Se utilizan señales *pulse-width modulation* (PWM) para el control de los *heaters* con el ciclo de trabajo devuelto por el algoritmo y en función de los datos procesados.

Los parámetros utilizados en la ejecución del algoritmo son Ttarget, temperatura objetivo para el lazo de control; C1 y C2, parámetros del lazo proporcional integral (PI), Plimit, potencia máxima comandable, Rheater, resistencia del *heater*; Rloss, resistencia de pérdidas del *heater* (cableado); V_heater, tensión de alimentación del *heater*; Tmax y Tmin, valores umbrales de temperatura para la detección de fallos; Tmeas_def, temperatura por defecto cuando empieza a funcionar la función; y Pmax_heater, potencia máxima entregada al *heater*.

La fórmula principal aplicada en la ejecución del algoritmo de control es:

$$P_{calc}(k) = P_{out}(k-1) + C_1 * \left(T_{target} - T_{meas}(k)\right) + C_2 * (T_{target} - T_{meas}(k-1))$$

(4.1)

Posteriormente, se realizan una serie de comparaciones para obtener los valores de la potencia de salida entregada al *heater* y el ciclo de trabajo.

La potencia máxima del *heater* la se calcula externamente con la siguiente fórmula:

$$Pmax_{heater} = \frac{R_{heater} * V_{heater}}{(R_{heater} + R_{loss})^2}$$

(4.2)

En los parámetros de configuración del sistema no están todos los anteriores, esto es debido a que la potencia máxima se calcula fuera de la FPGA, entonces los parámetros de resistencia del *heater*, tensión del *heater* y resistencia de pérdidas no se transmiten en los ficheros de configuración de la interfaz de comunicación.

Las especificaciones de la simulación del algoritmo en lazo cerrado se proporcionan con cálculos numéricos en una hoja de cálculo. [39]

5 DISEÑO

En el diseño e implementación se toman decisiones propias de diseño que no están definidas en las especificaciones técnicas del trabajo a realizar. [18], [39]–[51]

5.1 Materiales utilizados en el desarrollo.

Se elige trabajar con una *Field Programmable Gate Array* (FPGA) para poder realizar el diseño digital. Se decide trabajar con una FPGA Basys 3 de Xilinx, es una placa de desarrollo de circuitos digitales, tanto combinacionales como secuenciales. Se utiliza el software Vivado Design Suite para la síntesis y el análisis del diseño de los circuitos.

Fig. 10 Basys 3 [31]

La FPGA posee varios puertos USB y VGA, e incluye 16 switches, 16 LEDs y otros dispositivos de entrada y salida para poder realizar diseños *hardware* completos, sin añadir un hardware complementario. La FPGA tiene tres posibles modos de configuración, se utiliza el puerto USB-JTAG para configurarla y programarla. [41]

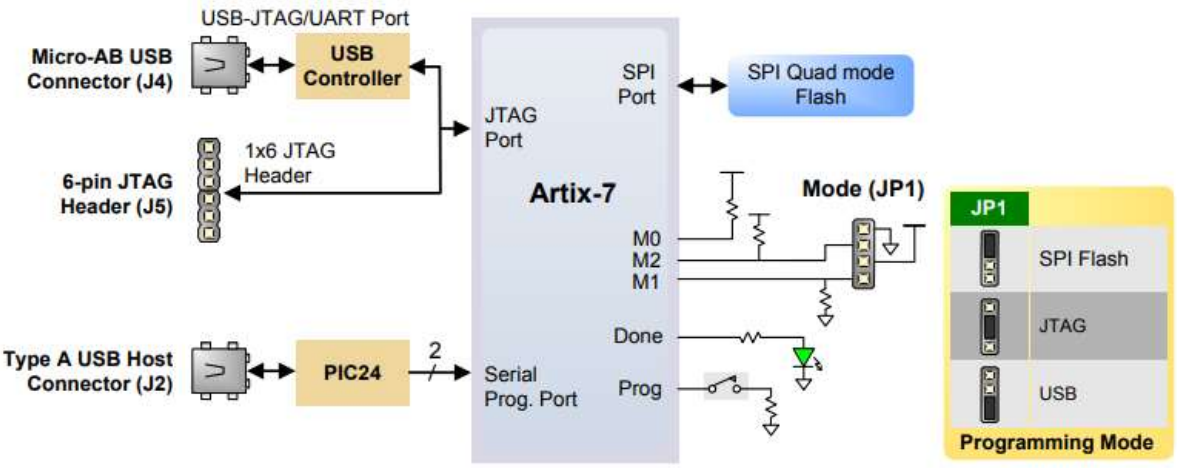


Fig. 11 Basys 3 opciones de configuración [41]

5.2 Adquisición de Temperatura.

El bloque de adquisición de temperaturas se encarga de adquirir los datos analógicos de las temperaturas de tres termistores externos. Los valores de temperatura se capturan consecutivamente hasta tener los datos de los tres termistores, se realiza una captura de los tres valores cada segundo.

La FPGA Basys 3 tiene un *Analog to Digital Converter* (ADC) interno que puede trabajar en diferentes modos, uno de ellos tiene la configuración que nos permite capturar los tres valores de manera sucesiva. [42] El ADC puede tener acceso a hasta 17 canales de entradas analógicas. En este bloque se usan tres de los canales disponibles, uno para cada termistor. Los datos analógicos de temperatura se digitalizan con el ADC interno, estos valores digitalizados tienen 12 bits de resolución. El ADC necesita una configuración específica para funcionar en el modo adecuado, la configuración definida para el modo de operación del ADC (*XADC Operating Mode*) es el modo secuencial de canales (*Automatic Channel Sequencer Mode*), se utiliza este modo para poder usar los tres canales del ADC necesarios para cada uno de los termistores, selecciona automáticamente el siguiente canal para la conversión, fija el tiempo necesario para la adquisición de datos y almacena los resultados en registros. El modo de tiempo (*Timing Mode*) es el modo de evento (*Event Mode*), se inicia la siguiente conversión cuando termine la conversión actual usando una entrada. El modo secuenciador (Sequencer Mode) es One-Pass, el secuenciador de canal selecciona el registro de operación y después

se detiene. La frecuencia del reloj interno del ADC es de 100MHz, y todo está sincronizado con el reloj DPR, sincronizado con el reloj interno de la FPGA. [10]

No se necesita un multiplexor externo para los canales del ADC, se utiliza el multiplexor interno que permite utilizar los tres canales seleccionados para la conversión.

El diagrama de bloques del ADC de la Basys 3 es: [42]

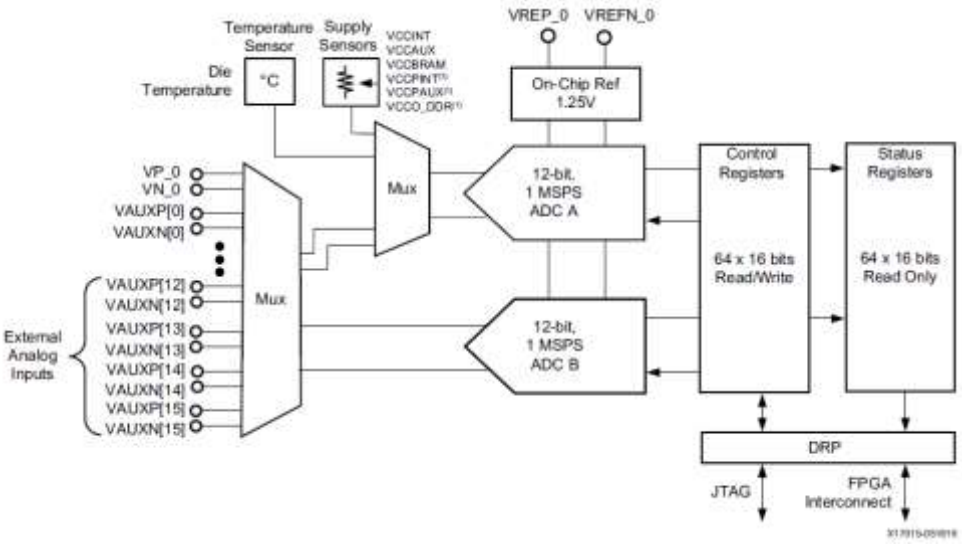


Fig. 12 Diagrama de bloques del XADC [39]

Todos los bloques y componentes del sistema que se van a describir tienen las señales de entrada del reloj global (Clk) y la señal para reiniciar el sistema y los biestables (Reset).

El diseño de bloques con las entradas y salidas realizado para el bloque de adquisición es el siguiente:

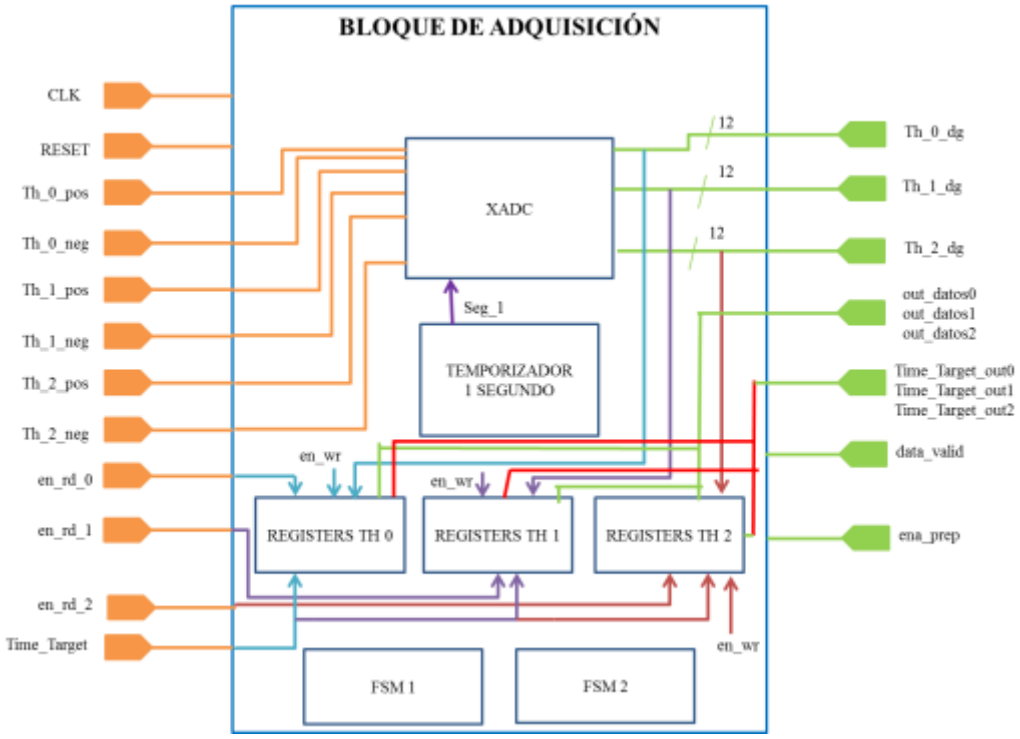


Fig. 13 Diagrama de bloques del bloque de adquisición

A continuación, se detallan cada uno de los bloques que se instancian como componentes en el bloque de adquisición.

5.2.1 Componente XADC.

El primero de ellos, es el ADC de la Basys 3, se le conoce con el nombre XADC. Este módulo convierte los datos analógicos capturados en datos digitales de 12 bits. Se utilizan tres de los 17 canales disponibles, uno para cada termistor. La obtención de los valores de temperatura se hace en modo bipolar, es decir, las entradas analógicas son diferenciales, entonces utilizan dos pines, uno para la entrada positiva y otro para la entrada negativa, este último se conecta a tierra (*Ground*, GND) si no se utiliza. Este módulo se instancia directamente en el diseño, y en él se definen las entradas analógicas. Al instanciar el XADC se puede acceder a todos los registros de estado de la conversión y a los resultados digitales de las medidas. El componente XADC con todos los puertos es el siguiente: [42]

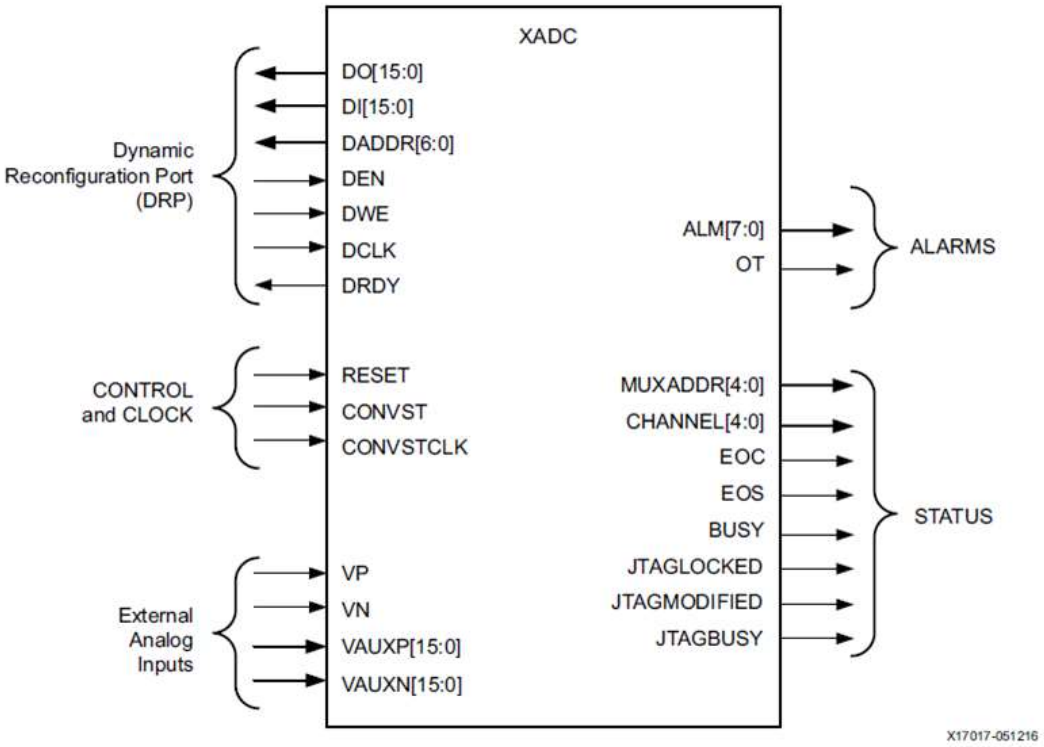


Fig. 14 Puertos del módulo XADC [42]

El componente instanciado en el diseño realizado y con la configuración mencionada anteriormente, tiene las siguientes entradas y salidas:

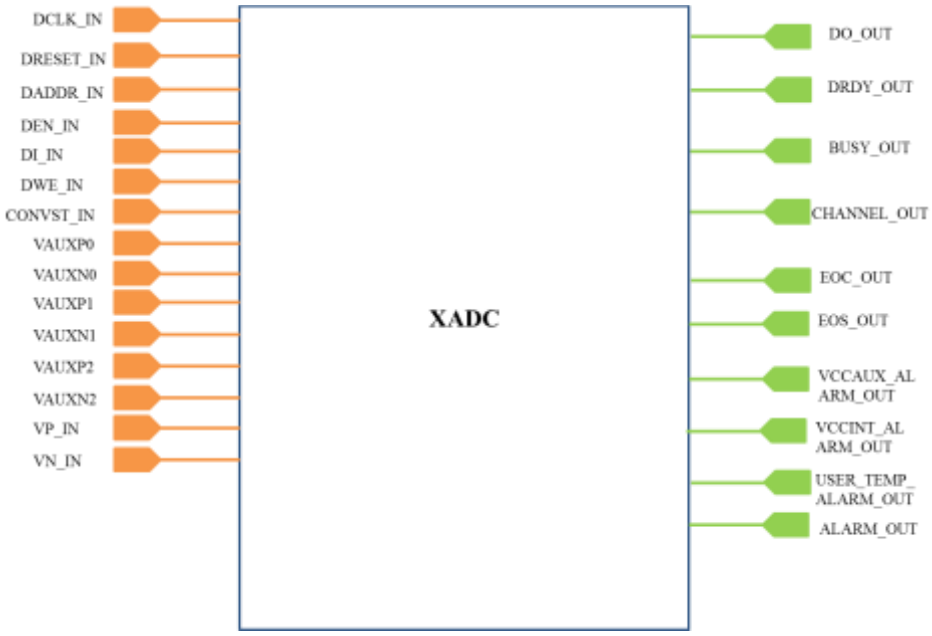


Fig. 15 Diagrama de entradas y salidas del componente XADC

5.2.2 Componente Registers.

El componente *Registers* se utiliza para almacenar los datos requeridos en las especificaciones. En el bloque de adquisición se guardan las últimas cinco muestras de temperatura de cada uno de los termistores, estos datos se guardan junto con el instante de tiempo en el que se capturan. Cada uno de los registros tiene una salida de datos y de la etiqueta de tiempo de los mismos del bloque de adquisición. Estas seis salidas, tres de valores de temperatura y tres de etiquetas de tiempo, se utilizan para poder leer los valores de los registros desde la UART.

Para guardar los valores convertidos por el ADC en el registro adecuado se utiliza un identificador, que se va actualizando según la activación en la segunda máquina de estados de la señal en_id.

Fig. 16 Diagrama de entradas y salidas del componente Registers

Para activar la señal de habilitación para escribir en los registros se utiliza un demultiplexor que depende del identificador de tabla, así solo se escribe en el registro correspondiente cada vez que la segunda máquina de estados activa la señal de habilitación de escritura (ena_wr).

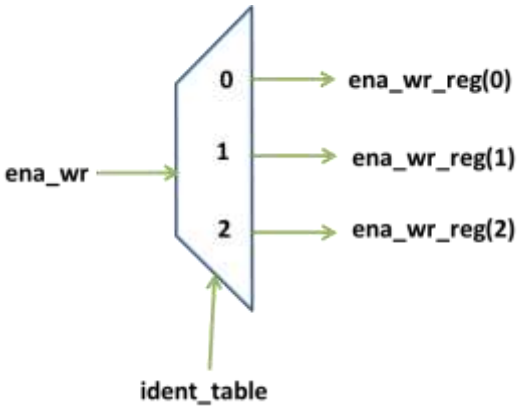


Fig. 17 Demultiplexor para las señales de habilitación de escritura

La lectura de los registros se realiza de manera combinacional cuando se activan las señales de habilitación de lectura desde la UART, la lectura se hace cuando se quieres mostrar los valores guardados en los registros y sus correspondientes etiquetas de tiempo.

5.2.3 Componente Temporizador.

Se utiliza en temporizador genérico. Es un contador ascendente con dos señales síncronas: *Clear* para poner el contador a cero y *Enable* para habilitar el contador, y una señal asíncrona, *Reset* fija el valor de la cuenta en caso de reiniciar el sistema. Este temporizador se utiliza para el lanzamiento de las conversiones de la secuencia de los canales, se realizan de forma consecutiva los tres canales, y se realiza cada segundo, entonces se necesita activar el conversor con un contador de 27 bits para poder contar 100.000.000 ciclos, equivalentes a 1 segundo de tiempo. El valor de la cuenta se va incrementando en cada flanco de subida del reloj y volverá a empezar cuando alcance el valor máximo de la cuenta, cuando se active la señal síncrona Clear o si se activa la señal asíncrona de Reset.

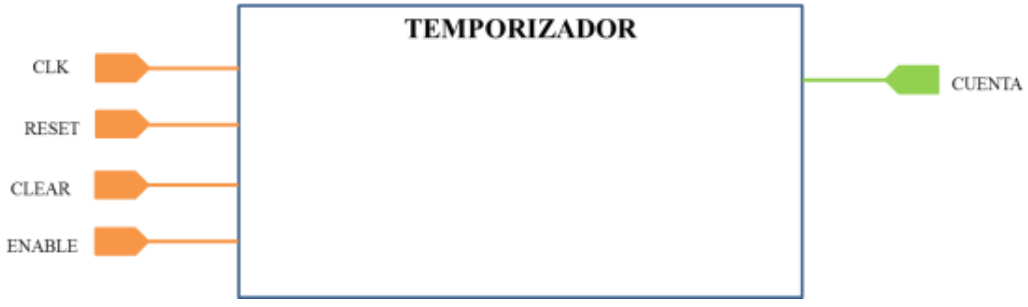


Fig. 18 Diagrama de entradas y salidas del componente Temporizador

5.2.4 Máquinas de estados.

Se utilizan dos máquinas de estados en el interior de este bloque.

La primera máquina se encarga del control del tiempo entre las diferentes adquisiciones de los tres termistores. Se adquieren de forma consecutiva los tres datos, y posteriormente se espera 1 segundo hasta que comienza la siguiente conversión de los tres termistores.

En el estado de reposo solo permanece un ciclo de reloj al inicio. Una vez en el estado de conversión se permanece ahí hasta que se haya terminado la conversión de los tres canales. En el estado siguiente se espera durante 1 segundo hasta que se inicia la siguiente conversión, y así sucesivamente. La señal de *running* se encarga de lanzar la conversión de la secuencia. La señal fin_sec se activa cuando termina la conversión de todos los canales de la secuencia a convertir, en este caso tres. La señal seg_1 se activa cuando el temporizador ha alcanzado el número de ciclos equivalente al paso de un segundo de tiempo. La señal del preprocesado (ena_prep) se encarga de indicar al bloque de preprocesado cuando ha terminado la conversión de la secuencia, se registran previamente los valores convertidos de los tres termistores capturados por el ADC. Cuando se activa la señal de ena_prep se realiza la votación por mayoría en el preprocesado y se guarda el resultado.

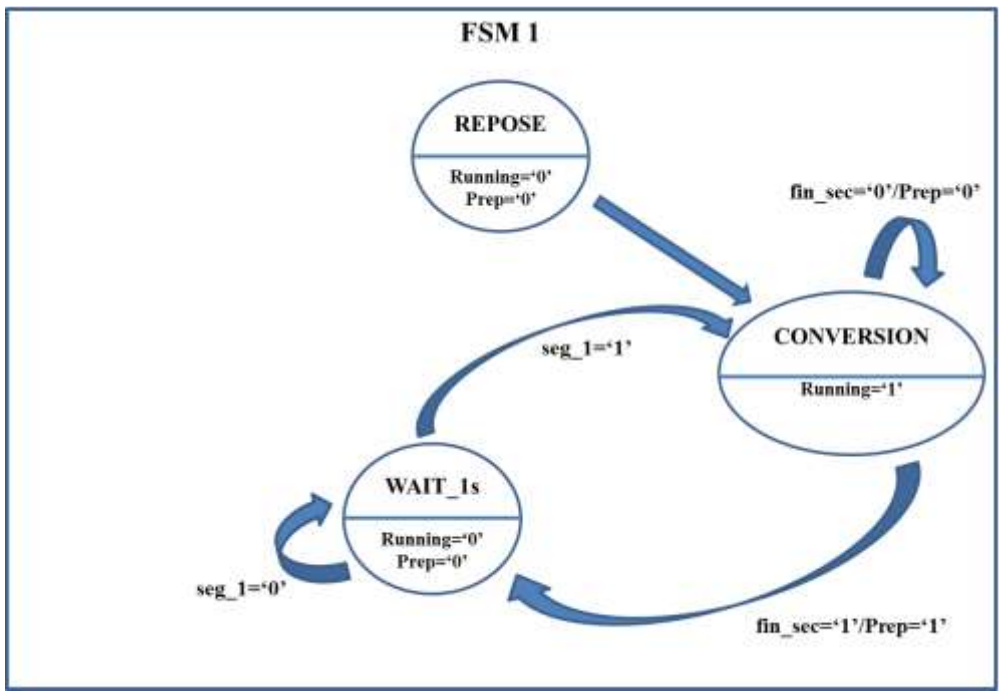


Fig. 19 Máquina de estados para controlar la conversión

Los estados de la máquina de control de la conversión son:

- REPOSE: Estado de reposo inicial.
- CONVERSION: Estado de conversión de toda la secuencia.
- Wait_1s: Estado de espera entre conversiones de secuencias diferentes.

La segunda máquina de estados se encarga del control de la conversión de cada canal del ADC.

En el estado de reposo permanece hasta que se activa la conversión de toda la secuencia, es decir, se activa la señal running. En el estado de empezar la conversión está un solo ciclo de reloj, en el que se activa la señal para iniciar la conversión, start. En el estado de espera de la conversión, se permanece hasta que hay un dato convertido, es decir, se activa la señal fin_conv y para indicar el fin de la conversión se activa la salida dato_conv, el resto permanecen inactivas en este estado. En siguiente estado estás hasta que el dato convertido está listo, se sabe que está disponible por el valor de la señal dato_rdy, se activa; las salidas están inactivas. Una vez que el dato está listo, se pasa al estado de cargar el dato, es decir, en este estado se guarda el valor convertido en el registro correspondiente, por eso, se activa la escritura en los registros, salida ena_wr, la salida dato_ok, que indica que el dato está disponible, y la salida en_id, para actualizar el identificador de los registros. Una vez se ha cargado el dato, si aún quedan datos de la misma secuencia por convertir se vuelve al estado de espera de la conversión, y si se han terminado de convertir los datos de los tres termistores se vuelve al reposo. Las salidas de esta máquina de estados son las siguientes: en_id, señal de habilitación para actualizar el identificador de los registros para guardar los datos en el registro correspondiente; dato_ok, señal que indica que el dato ya está listo para leerse; dato_conv, señal que indica que el dato ya está convertido aunque no está listo para leerse; start, se utiliza para indicar el comienzo de la conversión de la secuencia; y ena_wr, señal de habilitación de la escritura en el registro apropiado del dato ya disponible.

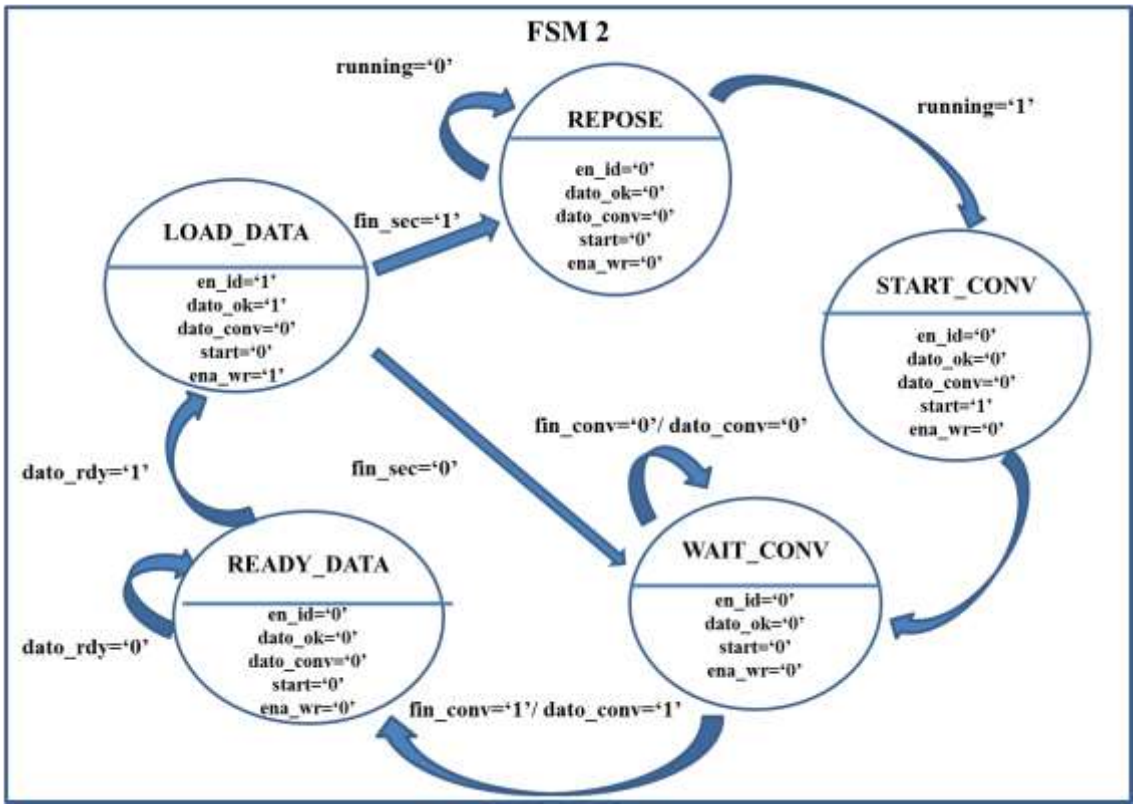


Fig. 20 Máquinas de estados para la conversión de todos los canales

Los estados de la máquina de estados de la conversión de todos los canales son:

- REPOSE: Estado de reposo hasta que comienza la conversión de la secuencia.
- STAR_CONV: Estado en el que comienza la conversión de la secuencia.
- WAIT_CONV: Estado de espera hasta que termina la conversión del canal.
- READY_DATA: Estado de espera hasta que el dato convertido está disponible.
- LOAD_DATA: Estado en el que se carga el dato convertido, y se almacena.

5.3 Gestión de tiempo (Time Management).

El bloque de gestión de tiempo se encarga de la sincronización del sistema completo.

La señal de entrada Sync_in, es una señal de sincronismo para cargar la etiqueta de Tiempo0 en la FPGA. El reloj interno de la FPGA es de 64 bits, y se configurará con la etiqueta de tiempo cuando se active la precarga con la señal de sincronismo externa. La etiqueta Time0 se configura externamente mediante la interfaz de comunicación con el usuario, y su resolución es de 1ms, es decir, cada milisegundo se modifica la etiqueta de tiempo de salida. La etiqueta de salida es la etiqueta de tiempo que se guarda junto con las muestras de los datos almacenados, esta etiqueta tendrá el valor del instante que corresponda.

Para la señal de sincronismo se utiliza un detector de flancos de subida, para evitar los rebotes del switch utilizado para la misma.

El bloque de gestión de tiempo tiene el siguiente esquema de entradas, salidas y componentes instanciados:

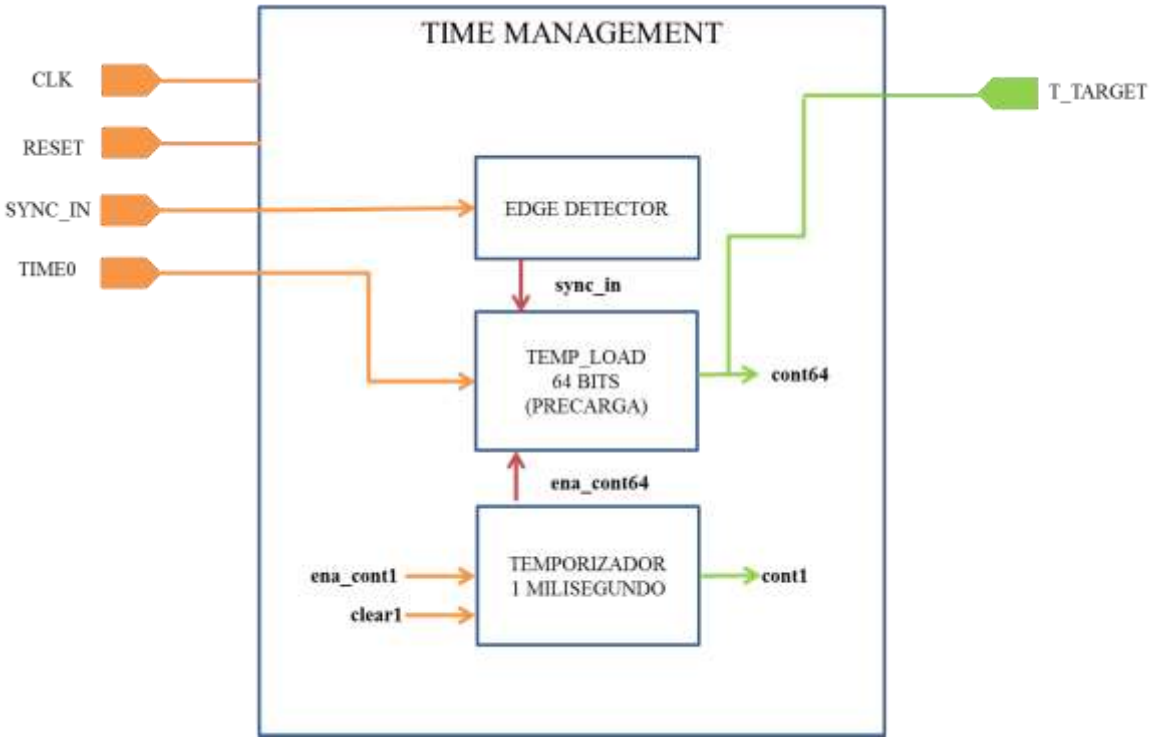


Fig. 21 Diagrama de bloques del bloque de gestión de tiempo (Time Management)

5.3.1 Componente Temporizador.

El temporizador instanciado es un temporizador genérico, es este caso tiene que contar un milisegundo. Para ello, tiene que ser de 17 bits, para ser capaz de contar 100.000 ciclos, que es el equivalente a 1ms. Cada vez que cuenta 1ms se encarga de activar el otro temporizador.

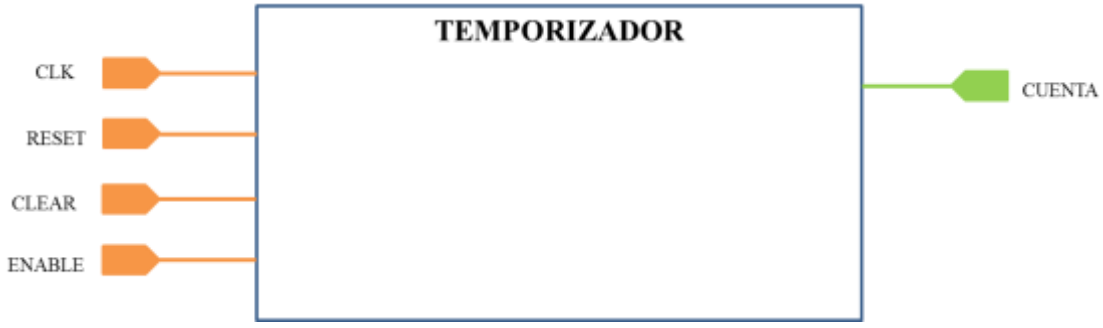


Fig. 22 Diagrama de entradas y salidas del componente Temporizador

5.3.2 Componente Temporizador con precarga.

El componente temp_load es un temporizador genérico con precarga. Es un contador de 64 bits. La precarga de este temporizador es la etiqueta de Tiempo0 que se configura a través de la interfaz de usuario, y se precarga cuando se active la señal de sincronismo. Los valores de la cuenta son instantes de tiempo y se guardaran junto a los valores almacenados en los registros cuando sea necesario.

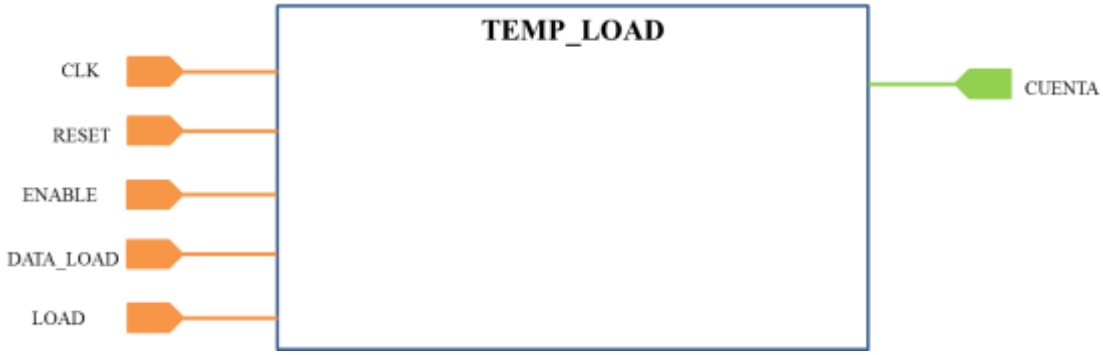


Fig. 23 Diagrama de entradas y salidas del componente Temp_Load

5.3.3 Componente Detector de Flancos.

Se utiliza un detector de flancos de subida en la entrada de sincronismo utilizada para la carga de la etiqueta de Tiempo0. Se utiliza el detector para evitar rebotes al activar la entrada.

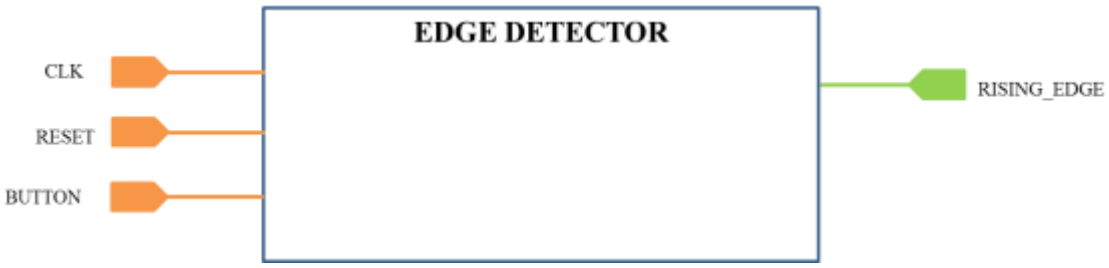


Fig. 24 Diagrama de entradas y salidas del componente Detector de Flancos

5.4 Interfaz de comunicación.

Se necesita una interfaz de comunicación para configurar la función de control térmico. A través de la interfaz se puede acceder a los parámetros de configuración y a los datos almacenados. Además, permite cargar y descargar los datos de las operaciones realizadas para desempeñar la función.

La interfaz de comunicación elegida para este diseño es la *Universal Asynchronous Receiver and Transmitter* (UART). UART es una interfaz de comunicación serie y la comunicación es full dúplex y asíncrona. Se utiliza para el intercambio de datos entre el PC y los periféricos de la FPGA. Habitualmente, se utiliza para el cambio de datos de escasa velocidad, coste y distancia entre PC y la placa. [52]–[56]

La UART tiene tres subcomponentes el módulo transmisor, el módulo receptor y el generador de la velocidad de transmisión, generador de baudios.

Hay dos canales uno para transmitir datos desde la FPGA al PC, y otro para recibir los datos del PC en la FPGA. En el canal de transmisión, se convierten los bytes que recibe la placa en paralelo en una secuencia de bits en serie que se transmite al PC. En el canal de recepción, el PC envía los bits en serie y se convierten en los bytes que el sistema es capaz de manejar. Cuando se transmite la

secuencia de bits en serie se añaden un bit de inicio (*start*) y un bit de fin (*stop*), y cuando se recibe la secuencia del PC estos dos bits se eliminan.

El bloque de control de la UART tiene las entradas y salidas mostradas en el diagrama. Está formado el componente de la UART y dos máquinas de estados, una para realizar la transmisión de datos, y otra para la recepción.

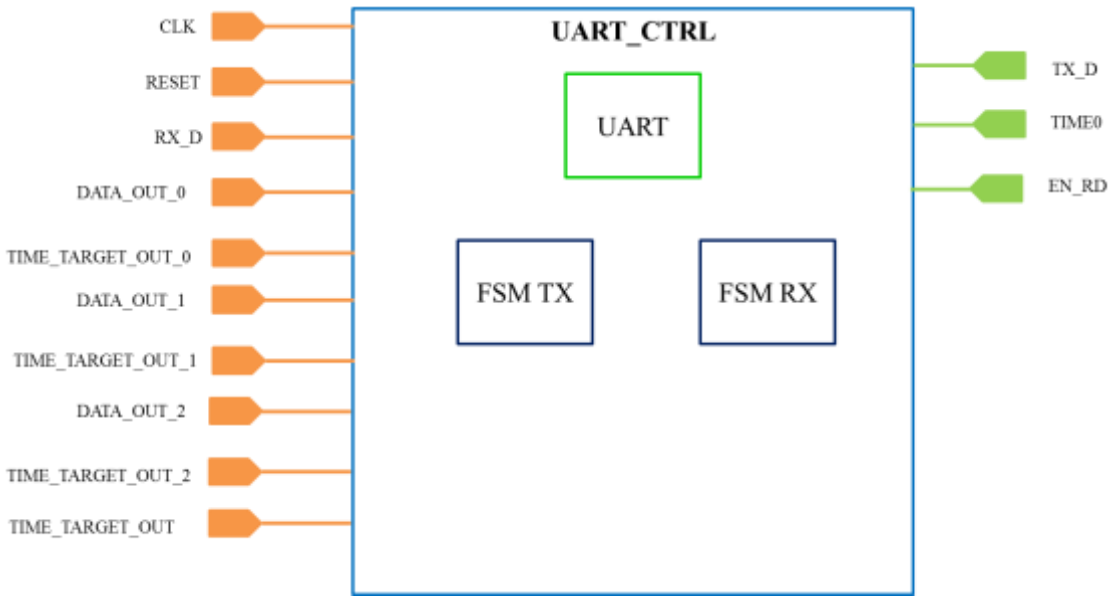


Fig. 25 Diagrama de bloques del bloque de control de la UART (UART_CTRL)

Las entradas de Datos_out y Time_target_out del bloque UART_CTRL se utilizan para mostrar los datos almacenados en el sistema: la temperatura de los tres termistores y la temperatura elegida por mayoría tras el preprocesado. Estos datos se mostrarán cuando se envíe uno de los comandos de recepción, el comando es la pulsación de la ‘r’ o ‘R’ en el teclado del ordenador.

5.4.1 Componente UART.

El componente de la UART que se instancia en el bloque de control cuenta con las siguientes entradas y salidas. Este bloque se encarga de la transmisión y la recepción de la interfaz UART, tiene la configuración necesaria.

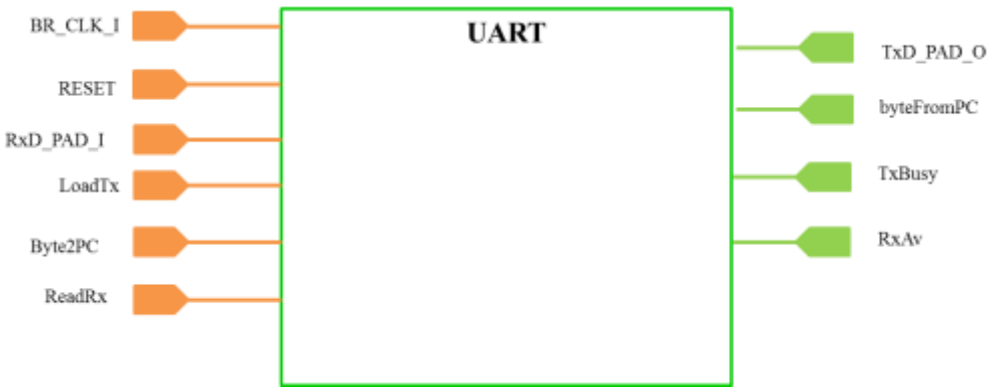


Fig. 26 Diagrama de entradas y salidas del componente UART

5.4.2 Máquinas de estados.

Las máquinas de estados de este bloque se utilizan para configurar los parámetros del sistema y para acceder a los datos almacenados. Se describen más detalladamente después.

5.4.2.1 Máquina de estados de Transmisión.

La máquina de estados para la transmisión se encarga de comunicar la FPGA con el PC. La FPGA transmite datos para que sean mostrados por el PC. Tiene dos opciones posibles, por un lado se muestran los parámetros de configuración del sistema, y por otro, los datos almacenados en los registros.

El estado inicial en ambos casos es INI_TX, en el que se permanece hasta que se recibe una ‘r’, ‘R’, ‘o’ u ‘O’. Si reciben la ‘r’ o la ‘R’ se pasa al estado de LOAD_DATA, es la primera opción, y si se reciben la ‘o’ o la ‘O’ se pasa al estado LOAD_DATA, la segunda opción.

Para la primera opción se utiliza la señal showReg, se activa en la máquina de estados de la recepción, cuando se pone a nivel alto se cambia del estado de inicio al de cargar los datos, LOAD_DATA. En este estado, se transmiten los diferentes registros de configuración al PC, dónde se muestran todos ellos. En el estado siguiente, REG_TX, se lleva una cuenta del número de registros que se han enviado ya al PC y del número de caracteres de cada registro para volver al estado inicial

cuando los 10 registros de configuración, con sus 14 dígitos cada uno, se han transmitido.

Para la segunda opción, se utiliza la señal showTable, se activa también en la máquina de estados de recepción, cuando se activa se cambia de estado. En el estado LOAD_TABLE, se transmite de la FPGA al PC una tabla con todos los datos guardados, es decir, con los valores de temperatura de los tres termistores de las últimas cinco muestras de cada uno de ellos, las últimas veinte muestras de Tmeas, las variables Tmeas y Dout de los últimos diez comandos de control térmico y los registros de las alarmas de sobrepaso. Se guarda el valor de la muestra junto con el instante de tiempo en el que son almacenados. En el siguiente estado, TAB_TX, se tiene un recuento del número de datos almacenados que se han transmitido, y del número de caracteres. Cuando se han transmitido los 25 registros con sus 50 caracteres cada uno se pasa al estado de inicio.

En el terminal virtual del PC, cuando se activa la señal showReg, se cargan los registros correspondientes, si no se han transmitido los valores de configuración tendrán un valor por defecto, pero no será mostrado en la pantalla. Se mostrará cuando se pulse ‘r’ o ‘R’ en el teclado del PC y se haya pasado un fichero.

En la imagen se puede observar el funcionamiento del terminal antes de tener una configuración cuando se escribe el comando ‘r’ o ‘R’ en el teclado.

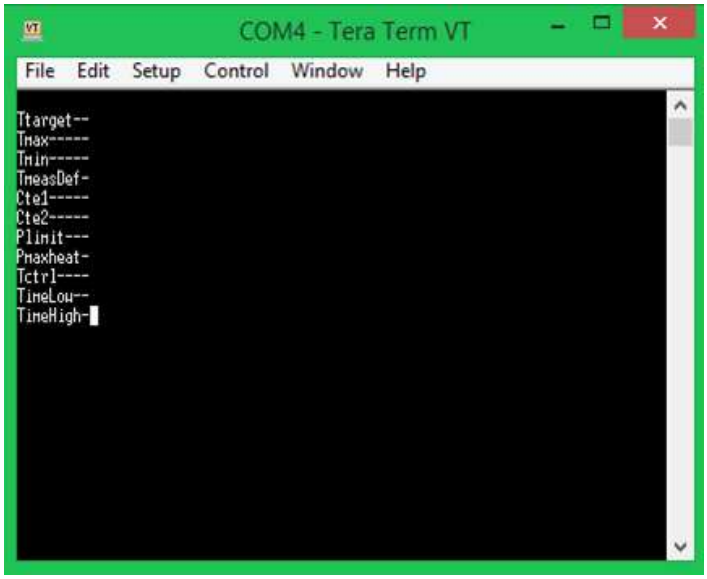


Fig. 27 Pantalla del terminal al recibir el comando ‘r’ o ‘R’

En el terminal virtual se muestra una tabla con todos los datos almacenados previamente en registros cuando se activa la señal showTable, es decir, cuando se recibe que se ha pulsado la tecla ‘o’ u ‘O’ en el teclado del PC, pero al principio, antes de tener valores en esos registros de almacenamiento.

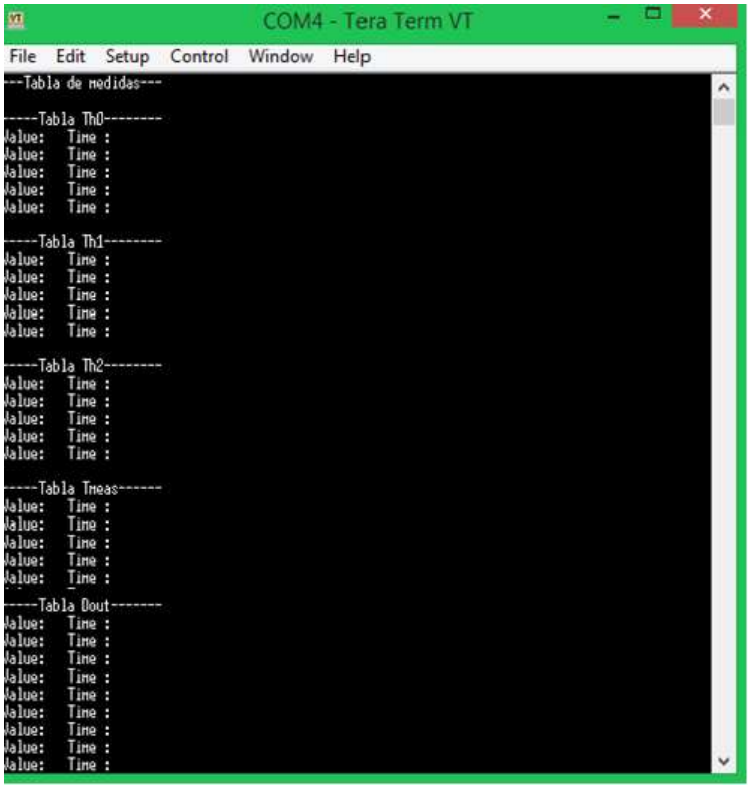


Fig. 28 Pantalla del terminal al recibir el comando ‘o’u ‘O’

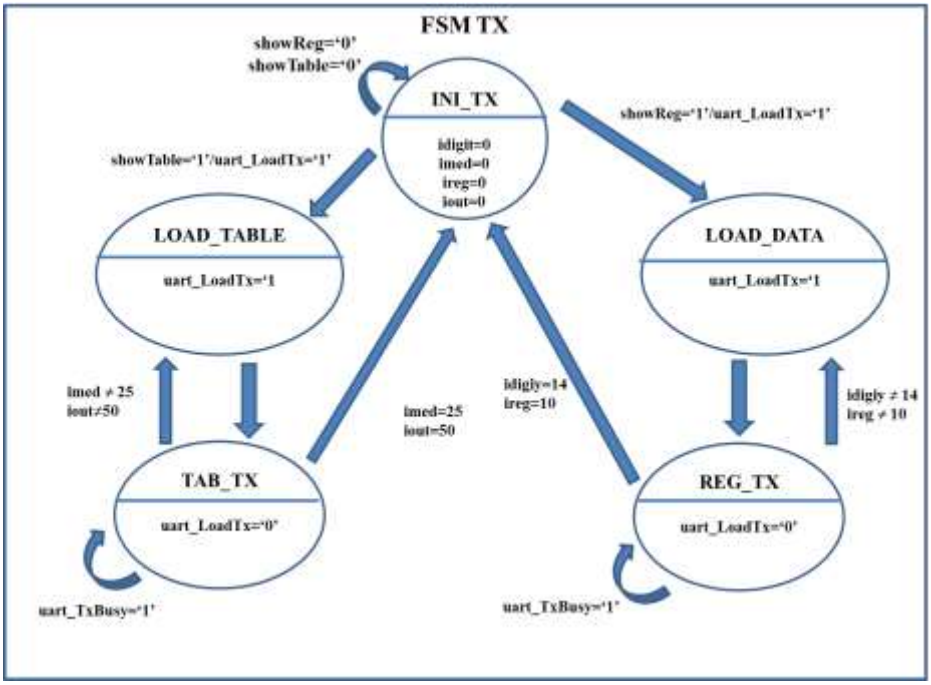


Fig. 29 Máquina de estados para la transmisión

Los estados de la máquina de estados para la transmisión son:

- INI_TX: Estado inicial hasta que se recibe un comando (r, R, o, O) en la máquina de recepción.
- LOAD_DATA: Estado para transmitir los datos de configuración de los registros.
- REG_TX: Estado en que se tiene en cuenta el número de registros transmitidos y el número de caracteres.
- LOAD_TABLE: Estado para transmitir los datos almacenados en los registros.
- TAB_TX: Estado en el que se tiene en cuenta el número de datos almacenados y el número de caracteres.

5.4.2.2 Máquina de estados de Recepción.

La máquina de estados para recepción se encarga de comunicar el PC con la FPGA. El PC transmite datos para que sean procesados en la FPGA. Se pueden recibir en la FPGA diferentes comandos que se escriben en el teclado del PC. Los comandos son 'r', 'R, 'o', 'O', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', ' space', 'CR' y 'LF' . Según el carácter recibido se pasa por unos estados u otros.

En el caso de recibir una 'r' o 'R', son dos comandos de lectura. Cuando se recibe una de las letras y a continuación un CR o un LF, se pasa del estado INI_RX al estado REG_SHOW, en el que se activa la señal showReg, usada en la máquina de estado de transmisión.

Si se recibe un '0', '1', '2', '3', '4', '5', '6', '7', '8', '9' o 'A' (número 10 en hexadecimal), se indica que es un identificador. Cada identificador se corresponde con uno de los parámetros de configuración. El '0' se corresponde con el parámetro de configuración T_{target} , el '1' con T_{max}, el '2' con T_{min}, el '3' con $T_{measDef}$, el '4' con Cte_1, el '5' con Cte_2, el '6' con P_{limit}, el '7' con $P_{maxheat}$, el '8' con T_{ctrl}, el '9' con $Time_{Low}$ y la 'A' con $Time_{High}$.

Una vez recibido el identificador se cambia al estado SPACE, en este estado se espera hasta que la FPGA recibe que se ha pulsado un espacio en el PC. Cuando se recibe el espacio se mueve al estado VALUE.

En el estado VALUE, se espera hasta que se recibe un retorno de carro (CR) o fin de línea (LF). En este estado se recibe los valores de configuración de cada uno de los parámetros. Cuando ya se ha recibido el valor del parámetro que se está configurando se recibe un CR o un LF y se vuelve al estado inicial.

Para la configuración de cada uno de los parámetros se pasa por los tres estados: INI_RX, SPACE y VALUE.

En el caso de recibir una 'o' o una 'O', son dos comandos para mostrar los valores de los datos almacenados. Cuando se recibe una de las letras y a continuación un CR o un LF, se pasa del estado INI_RX al estado REG_TABLE y se activa la señal showTable, usada en la máquina de estado de transmisión.

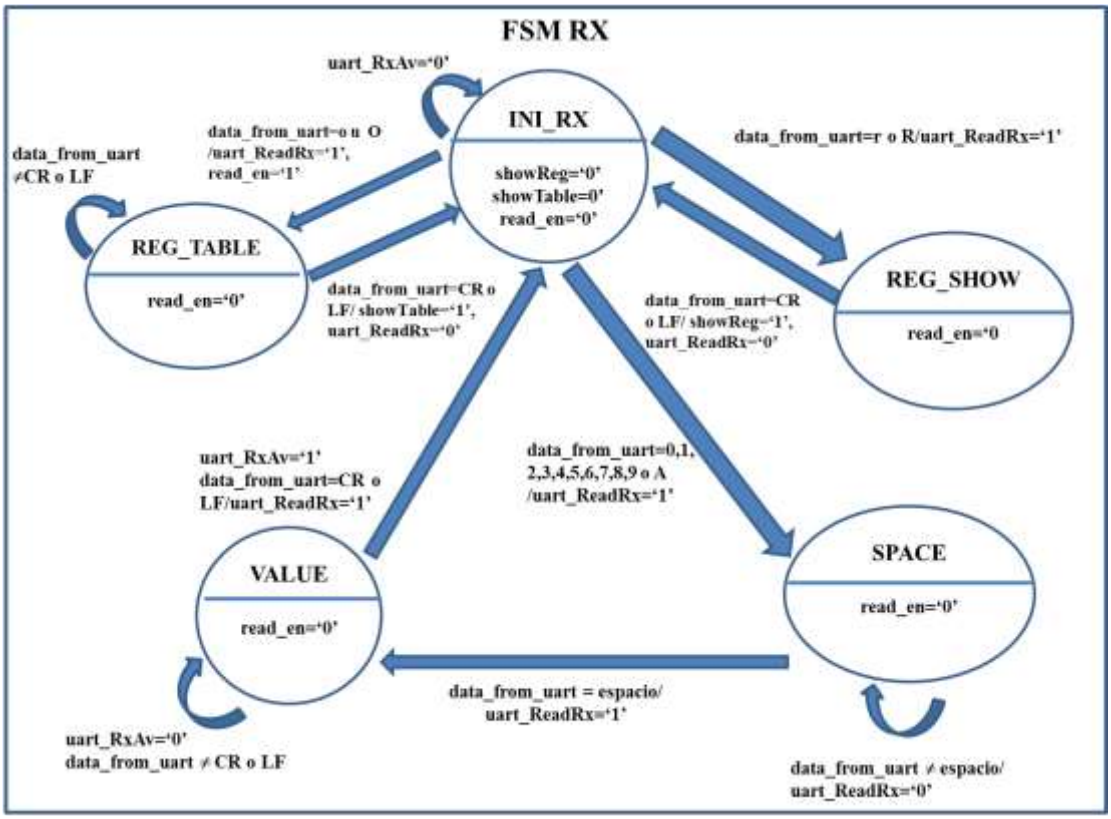


Fig. 30 Máquina de estados para la recepción

Los estados de la máquina de estados para la recepción son:

- INI_RX: Estado de reposo hasta que se recibe uno de los comandos (r, R, o, O).
- REG_SHOW: Estado para activar la señal de la máquina de transmisión para los datos de configuración.

- ❖ SPACE: Estado de espera para recibir un espacio tras el identificador.
- ❖ VALUE: Estado de espera para recibir los valores de configuración.
- ❖ REG_TABLE: Estado para activar la señal de la máquina de transmisión para los datos almacenados.

5.5 Algoritmo de Control.

El algoritmo de control se ejecuta para calcular la potencia entregada al *heater* y para calcular el ciclo de trabajo (*Duty Cycle*) necesario para comandarlo.

El control térmico se encarga de calcular el *Duty Cycle* de la potencia que se deja pasar al *heater*, la potencia entregada es controlada con un interruptor. En este control, se generan las señales para comandar al *heater*.

El bloque del algoritmo de control con las entradas y salidas y los componentes, se muestra a continuación.

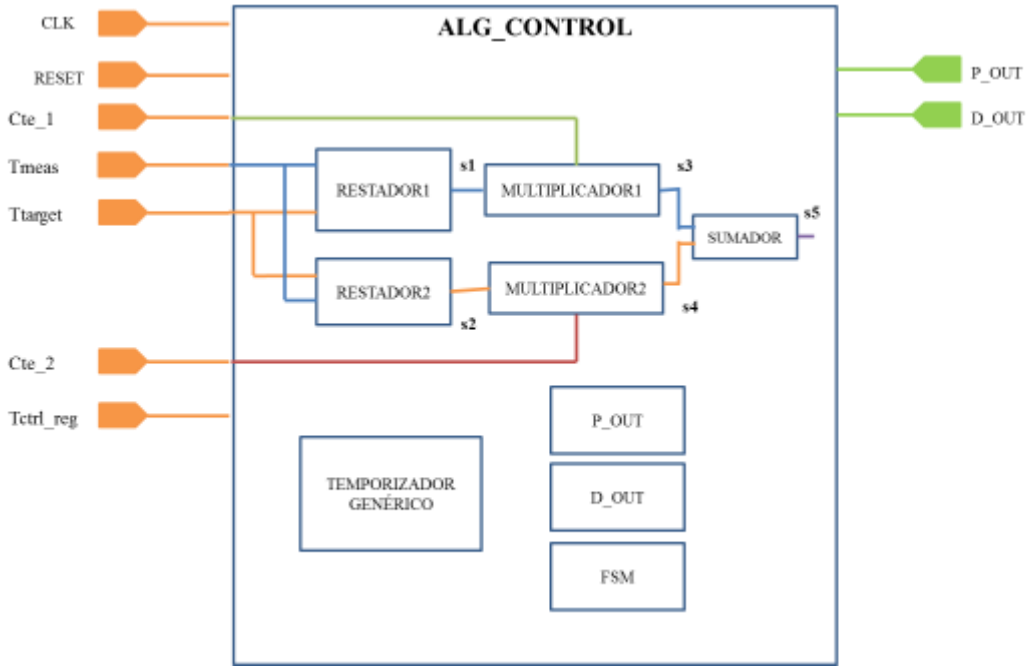


Fig. 31 Diagrama de bloques del bloque del Algoritmo de Control

En la ejecución del algoritmo de control se realizan cálculos de manera combinacional, es decir, se están haciendo continuamente. Estos valores calculados se utilizan para poder obtener la potencia de salida entregada al *heater* y el ciclo de trabajo. Se necesitan dos restadores, para uno se utiliza el valor de T_{meas} en ese

instante y T_{target}, y para el otro el valor de T_{meas} en el instante anterior y T_{target}; y dos multiplicadores, de los resultados parciales anteriores con las constantes 1 y 2. Con estas operaciones se obtiene el resultado interno s5. El valor T_{meas} se calcula al aplicar el algoritmo con el valor del instante anterior, solo se define el primer valor para poder inicializar la ejecución del algoritmo.

Se diseña el circuito para aplicar la ecuación del algoritmo:

$$P_{calc}(k) = P_{out}(k-1) + C_1 * \Big(T_{target} - T_{meas}(k)\Big) + C_2 * (T_{target} - T_{meas}(k-1))$$

(5.1)

A continuación se calcula potencia de salida entregada al *heater*:

$$P_{out}(k) = \begin{cases} P_{min} & si\ P_{calc}(k) < P_{min} \\ P_{calc}(k) & si\ P_{min} < P_{calc}(k) < P_{max} \\ P_{max} & si\ P_{calc}(k) > P_{max} \end{cases}$$

(5.2)

Por último, se calcula el ciclo de trabajo para comandar al *heater*:

$$D(k) = \begin{cases} 255 & si\ P_{out}(k) > P_{\max\_heater} \\ ENTERO\left(\frac{255 * P_{out}(k)}{P_{\max\_heater}}\right) & si\ P_{out}(k) \leq P_{\max\_heater} \end{cases}$$

(5.3)

Las temperaturas (T_{target}, T_{meas}) son señales en coma fija de 12bits, 6 bits de parte entera, 5 bits de parte decimal y 1 bit de signo.

Las constantes temperaturas son señales en coma fija de 10bits, 1 bit de parte entera, 8 de parte decimal y 1 bit de signo.

La potencia de salida es una señal en coma fija de 13 bits, 3 de parte entera, 9 de parte decimal y 1 bit de signo.

El ciclo de trabajo es un número entero.

5.5.1 Componente Temporizador

El temporizador genérico es un componente de este bloque. El algoritmo de control se ejecuta periódicamente cuando el temporizador ha alcanzo el número de ciclos indicados por el período de control, que es una entrada del bloque, T_{ctrl_reg}. Este componente indica cuando se tiene que empezar a ejecutar el algoritmo, no se realiza siempre pasado el mismo tiempo, por su dependencia de la entrada de control.

Fig. 32 Diagrama de entradas y salidas del componente Temporizador

5.5.2 Máquina de estados.

En este bloque hay una máquina de estados que controla la ejecución del algoritmo de control.

Se permanece en el estado de reposo, IDDLE, hasta que el temporizador ha contado el tiempo correspondiente al período de control, entonces se empieza a ejecutar. En el estado STEP 1 permanece un ciclo de reloj. En este estado, se registra el valor de todas las entradas con las que se realizan operaciones durante la ejecución del algoritmo. Luego se cambia al estado STEP 2, donde se permanece durante un ciclo de reloj. En este estado, se realiza la operación para calcular la P_calc, es la potencia que servirá para fijar la potencia de salida entregada al *heater*. El valor de esta potencia queda guardado en un registro para poder utilizarlo.

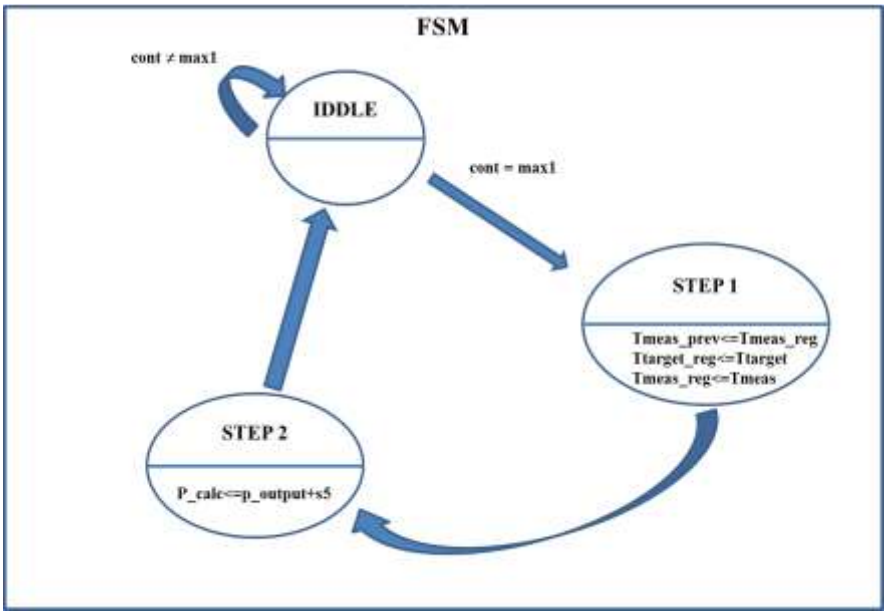


Fig. 33 Máquina de estados para la ejecución del Algoritmo de Control

Los estados de la máquina de estados para la ejecución del Algoritmo de control son:

- IDDLE: Estado de reposo hasta que se ha temporizado el tiempo correspondiente al período de control.
- STEP1: Estado en el que se registra el valor de las entradas.
- STEP2: Estado en el que se calcula la potencia calculada.

5.5.3 Potencia de salida.

Para calcular la potencia de salida entregada al *heater* (P_output), se realizan una serie de operaciones con valores obtenidos en cálculos anteriores.

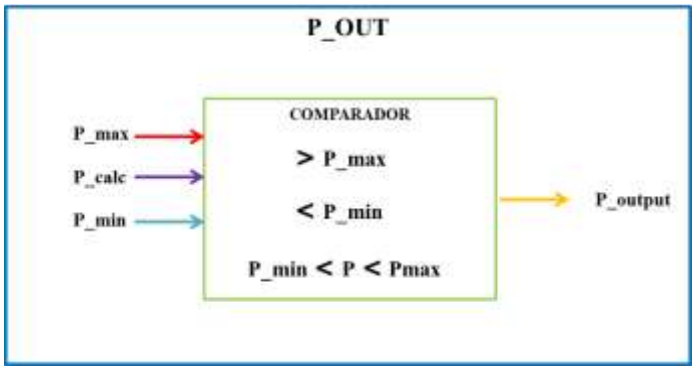


Fig. 34 Comparador para calcular la potencia de salida

La potencia de salida se obtiene tras realizar tres comparaciones o menos. Si el valor de la potencia calculada en la ejecución de la máquina de estados, P_calc, es mayor que un valor de potencia máxima fijado, entonces el valor de la potencia de salida satura al valor de la potencia máxima. Si el valor de P_calc es menor que el de la potencia mínima con la que se compara, entonces P_output tendrá el valor de esa potencia mínima. Por último, si la potencia calculada tiene un valor entre medias de la potencia máxima y la potencia mínima, entonces la potencia de salida toma el valor de la potencia calculada.

5.5.4 Ciclo de trabajo de salida.

Al igual que para calcular la potencia de salida, se utilizan comparaciones para el ciclo de trabajo de salida.

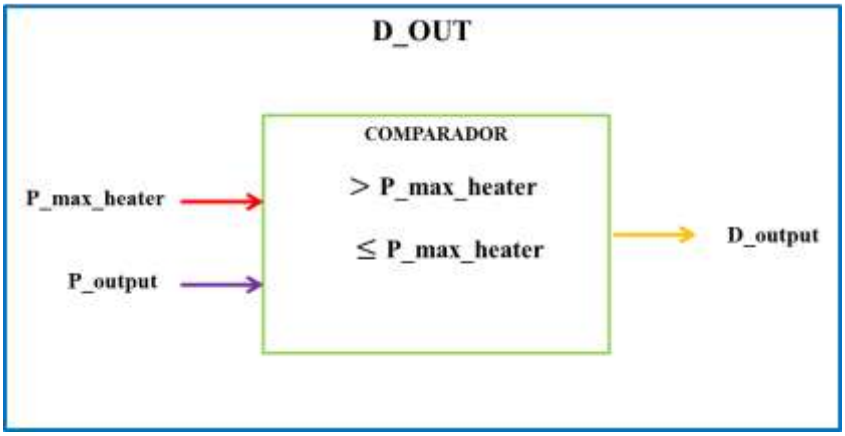


Fig. 35 Comparador para calcular el ciclo de trabajo de salida

El ciclo de trabajo se calcula realizando comparaciones entre la potencia de salida calculada anteriormente y la potencia máxima que puede tener el *heater*. Si la potencia de salida es mayor que la potencia del *heater*, entonces el ciclo de trabajo tendrá un valor de 255, en caso contrario se realizará una operación para obtenerlo. La operación realizada es multiplicar 255 por una fracción $\frac{P\_output}{P\_\max\_heater}$.

5.6 Preprocesado.

El bloque del preprocesado se encarga de determinar la temperatura media de los termistores, T_{meas}.

Para definir la temperatura media se define por un sistema de votación por mayoría.

El sistema de votación definido está basado en tres comparadores de igualdad. Si los valores de temperatura de los termistores 0 y 1 son iguales entonces el valor de T_{meas} es el del termistor 0. Si los valores de temperatura de los termistores 0 y 2 son iguales, el valor de T_{meas} será el del termistor 2. Si los valores de temperatura de los termistores 1 y 2 son iguales, el valor de T_{meas} será el del termistor 1. El valor de termistor 0 tiene prioridad sobre el resto, entonces en caso de que varias de las comparaciones sean ciertas o si ninguna igualdad es cierta el valor de T_{meas} estará fijado por el dato del termistor 0.

Este bloque tiene el componente *Registers* para almacenar los datos de T_{meas} una vez preprocesados. Se almacenan las últimas veinte muestras de la temperatura junto con el instante de tiempo en el que son guardados. Los datos se guardan cada vez que activa la entrada Ena_prep aunque el preprocesado se está realizando continuamente con los valores que llegan solo se guardan los valores en los instantes concretos. Esta entrada es activada en el bloque de adquisición de temperaturas cuando se terminan todas las conversiones de la secuencia. Los datos guardados en los registros es el valor de T_{meas} junto con el instante de tiempo en el que se almacena el dato, el valor de de T_{meas} es el calculado con la votación por mayoría.

Las entradas del bloque T_{max} y T_{min} son los parámetros configurables a través de la UART. La temperatura calculada por votación T_{meas} se compara con estos rangos de temperatura, si la temperatura Tmeas es superior a T_{max} o inferior a T_{min}, se activa la alarma para detectar el fallo.

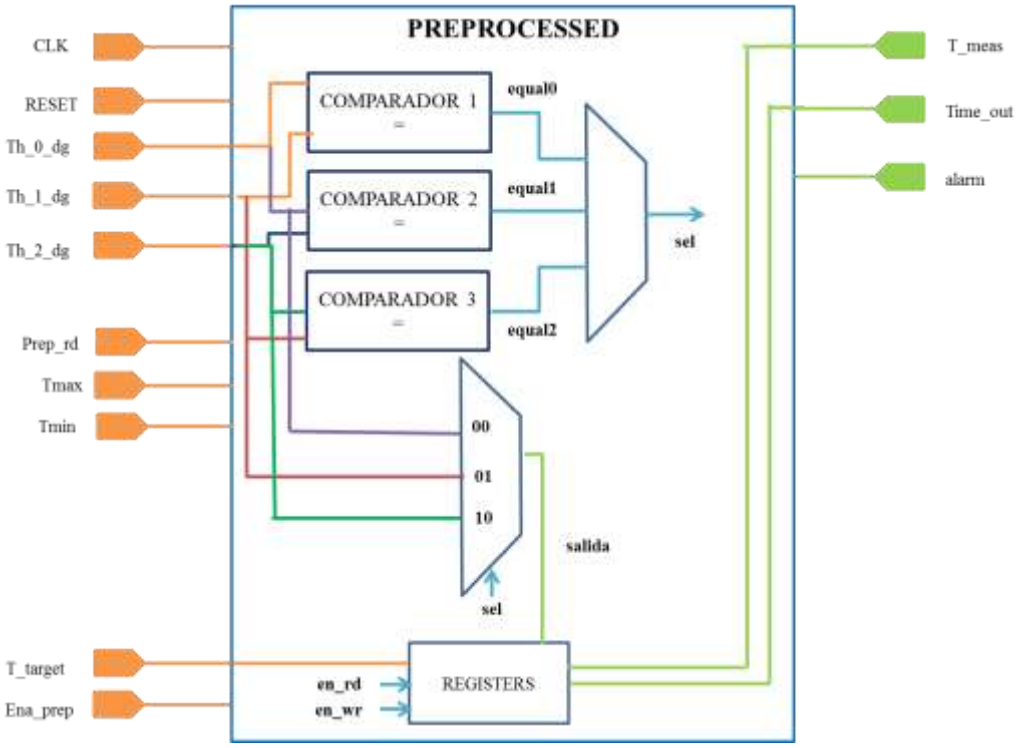


Fig. 36 Diagrama de bloques del bloque del preprocesado

5.7 Remote Terminal Unit.

Este bloque es el encargado de conectar todos los bloques descritos anteriormente y que forman parte del sistema.

Para conectar todos los componentes se utilizan señales internas, que unen las salidas de unos bloques con las entradas de otros.

Las conexiones establecidas entre los bloques se encargan del control del sistema, para ello se utilizan señales que habilitan el funcionamiento de otros bloques diferentes al que genera esa señal.

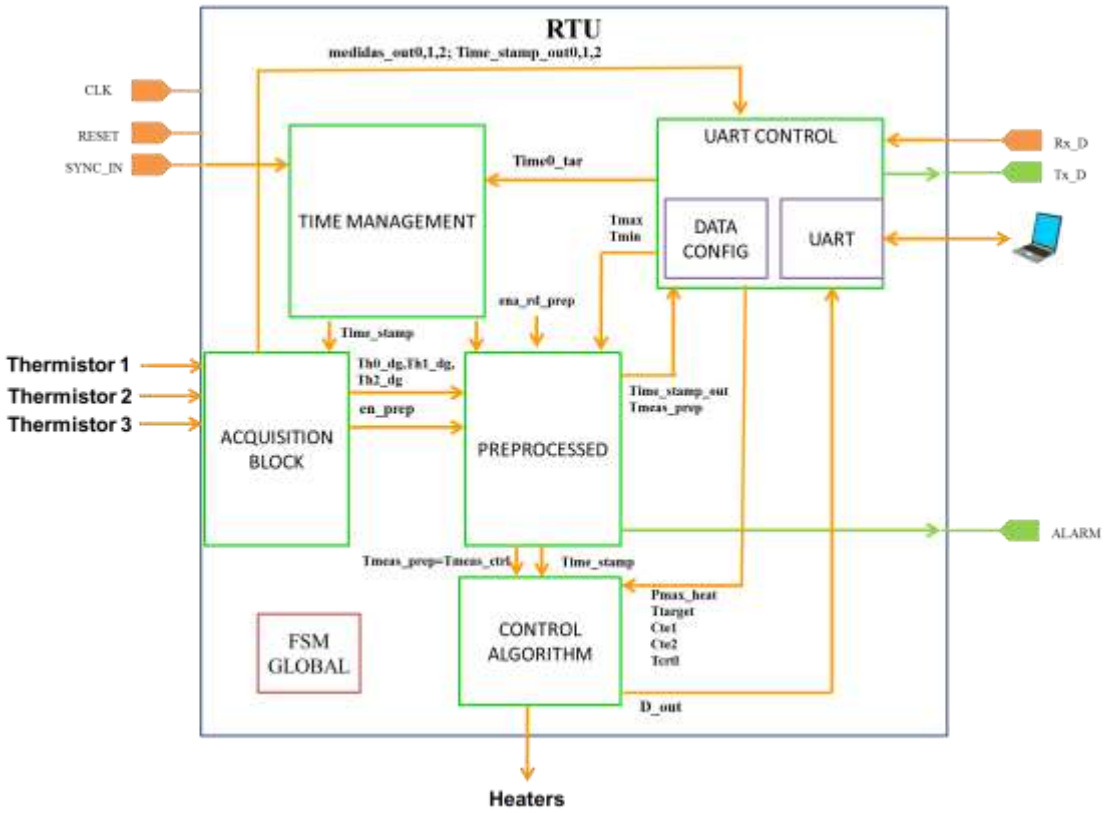


Fig. 37 Diagrama de bloques del sistema completo

5.7.1 Máquina de estados.

En este bloque se utiliza una máquina de estados para controlar el sistema de manera global, para el correcto funcionamiento de todos los componentes que se instancian en el bloque RTU.

La máquina de estados de este bloque está diseñada para controlar el componente de los registros que almacena datos, es decir, del componente Registers del bloque de adquisición y del bloque del preprocesado.

Tiene solamente dos estados, uno de reposo, IDDLE, y otro de habilitación de las señales de lectura, READING. En el estado de reposo permanece hasta que la señal regis_rd se activa. Esta señal es una salida del bloque UART_CTRL, y se activa cuando en el teclado del ordenador se pulsa una ‘o’ o una ‘O’, es decir, cuando se quieren mostrar por pantalla todos los datos almacenados en los registros: los últimos cinco valores de temperatura de cada uno de los termistores, las últimas veinte muestras de T_{meas} y los últimos diez valores del ciclo de trabajo

de salida, D_out. En el estado de lectura permanece un solo ciclo de reloj, en el que se habilitan todas las salidas de la máquina de estados. Las salidas son en_0_rd, en_1_rd, en_2_rd y en_rd_prep, las tres primeras son entradas del bloque de adquisición, y habilitan las señales de lectura de los registros instanciados como componentes en el bloque de adquisición, para poder leer los datos almacenados en los registros desde la UART. La salida en_rd_prep habilita la entrada de lectura del componente de registros instanciado en el bloque de preprocesado, con la misma finalidad que las otras tres señales, para poder acceder a los valores guardados y transmitirlos por la UART.

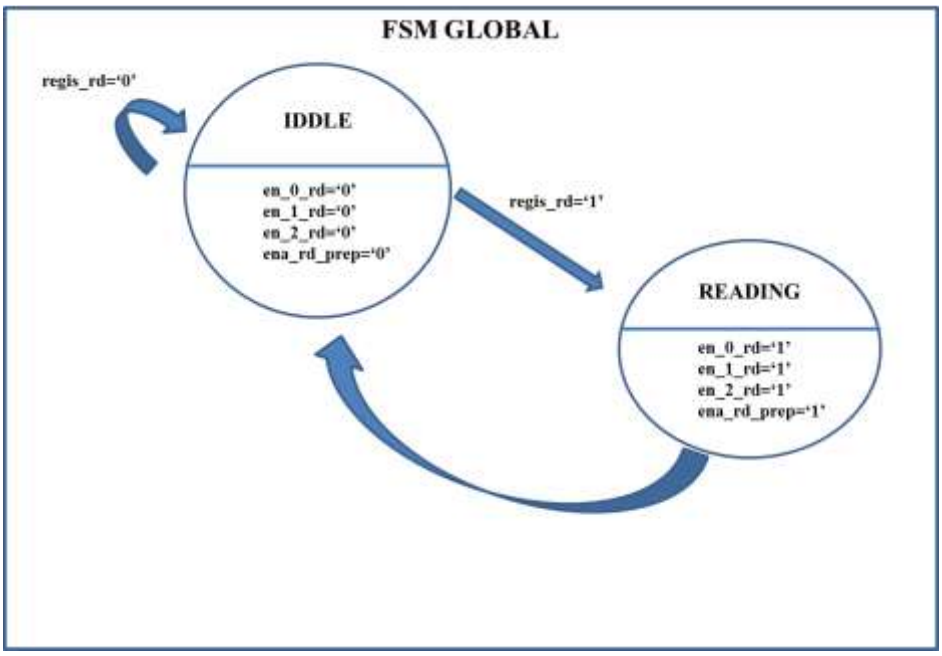


Fig. 38 Máquina de estados para el control del sistema

Los estados de la máquina de estados para el control del sistema son:

- IDDLE: Estado de reposo hasta que se recibe el comando de habilitación de lectura de los registros.
- READING: Estado para activar las señales de habilitación de lectura de los registros.

6 VALIDACIÓN Y PRUEBAS EXPERIMENTALES

En este apartado se van a detallar las pruebas realizadas para la validación del diseño realizado. De esta forma se comprueba el correcto funcionamiento del sistema y de cada uno de los bloques que forman parte del sistema completo.

Se realizan pruebas de diferentes tipos: pruebas en simulación, pruebas en la FPGA y pruebas en hojas de cálculo.

Para realizar las pruebas de simulación se utiliza la herramienta de simulación del software de Vivado, Vivado® Simulator. Permite hacer simulaciones en lenguaje el lenguaje de descripción hardware VHDL sin limitación en el tamaño de los diseños, número de instancias o de ejecuciones.

Para las pruebas con la FPGA se realizan montajes hardware externos adicionales a la FPGA en alguna de ellas.

Las pruebas se hacen para comprobar el funcionamiento de los diferentes componentes o bloques que forman el sistema completo.

6.1 Pruebas realizadas en FPGA.

Las pruebas realizadas en la FPGA se detallan en los siguientes apartados.

6.1.1 Prueba del XADC.

Para la realización de esta comprobación se utiliza la FPGA y adicionalmente se utiliza un circuito externo.

Se realiza la conversión de un solo canal del XADC, en este caso se utilizaba el canal Vaux6. La configuración del XADC para la realización de la prueba es *Channel Sequencer*, se selecciona automáticamente el siguiente canal para la conversión, en este caso solo se puede elegir el canal utilizado; *Event Mode*, la conversión termina cuando se activa el bit de parada y se inicia la siguiente usando una entrada y *One Pass*, el secuenciador de canal selecciona el registro de operación y después se detiene.

Se realiza un circuito divisor de tensión con dos resistencias una de 10KΩ, R_1, y otra de 10KΩ, R_2, se conecta a una fuente de tensión variable.

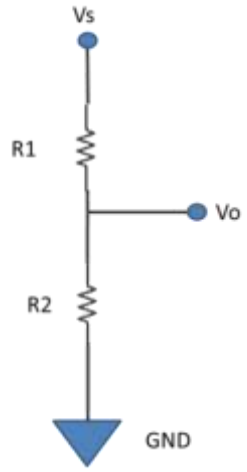


Fig. 39 Circuito divisor de tensión

Se realiza esta prueba para comprobar el correcto funcionamiento del conversor analógico digital, del módulo XADC. La tensión de la fuente se va variando desde 0V hasta un máximo de 3.3V, que es la tensión máxima que se puede medir con el XADC. Se mide la tensión Vo con un polímetro y se calcula el valor digital que se corresponde con el valor de tensión medido.

Para calcular la tensión Vo se aplica la fórmula del divisor de tensión, para los diferentes valores de la fuente que se va variando.

$$Vo = Vs \cdot \frac{R2}{R2 + R1} \tag{6.1}$$

Después, se calcula el valor digital que se corresponde con esa tensión.

$$D = Vo \cdot \frac{2^{12} - 1}{Vs} \tag{6.2}$$

A continuación, se comprueba que el valor digital calculado numéricamente se corresponde con el valor devuelto por el ADC. La salida del ADC se muestra por los LEDs de la FPGA.

Tras probar varios valores de tensión se puede verificar que el módulo del conversor analógico digital funciona correctamente.

En el montaje había una fuente de alimentación que se variaba desde 0V a 3.3V, en la FPGA se observan que los LEDs encendidos varían en función de la tensión. Al mismo tiempo se medía la tensión en la resistencia R2 con el polímetro para controlar que el valor digital devuelto en la FPGA, se correspondía con la tensión medida por el polímetro.

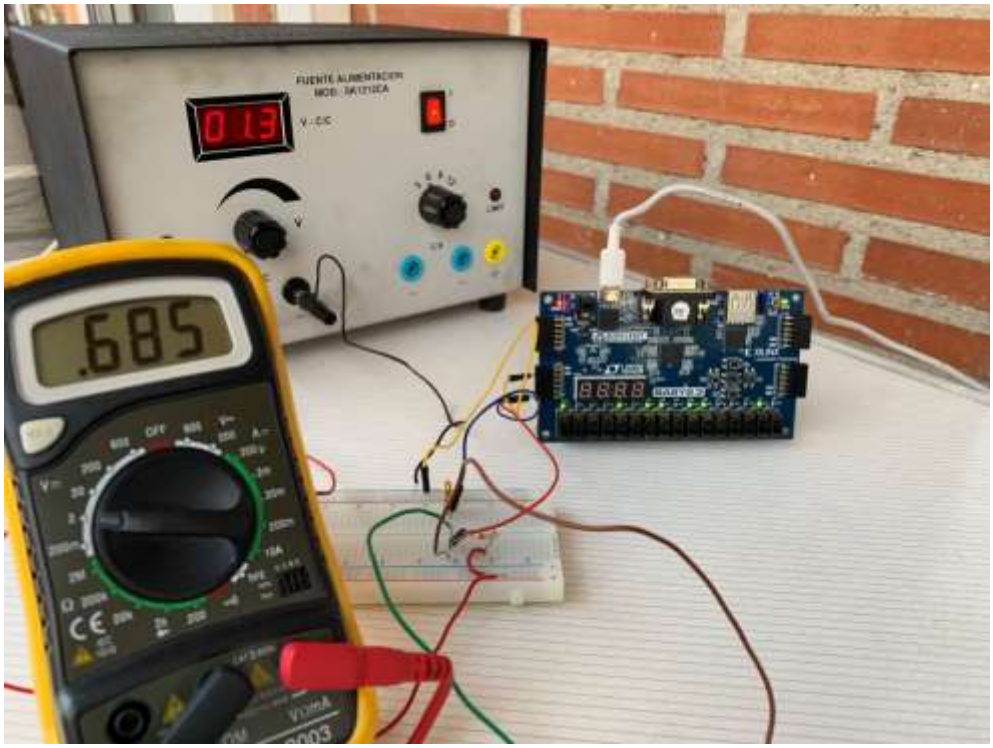


Fig. 40 Montaje para una tensión intermedia

En las dos imágenes mostradas a continuación se muestran los resultados para las tensiones máxima y mínima del rango permitido.

En este montaje se puede observar que todos los LEDs de la FPGA están encendidos que se corresponde con la saturación máxima del ADC.

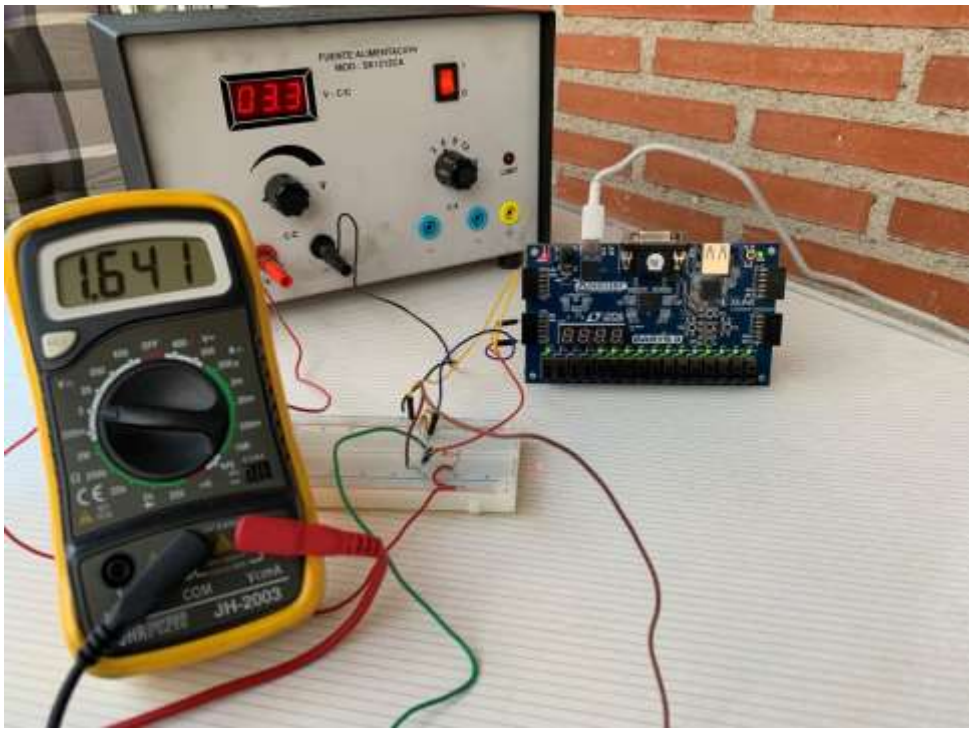


Fig. 41 Montaje para la tensión máxima

En este montaje se observa que todos los LEDs de la FPGA están apagados que se corresponde con la saturación mínima del ADC.

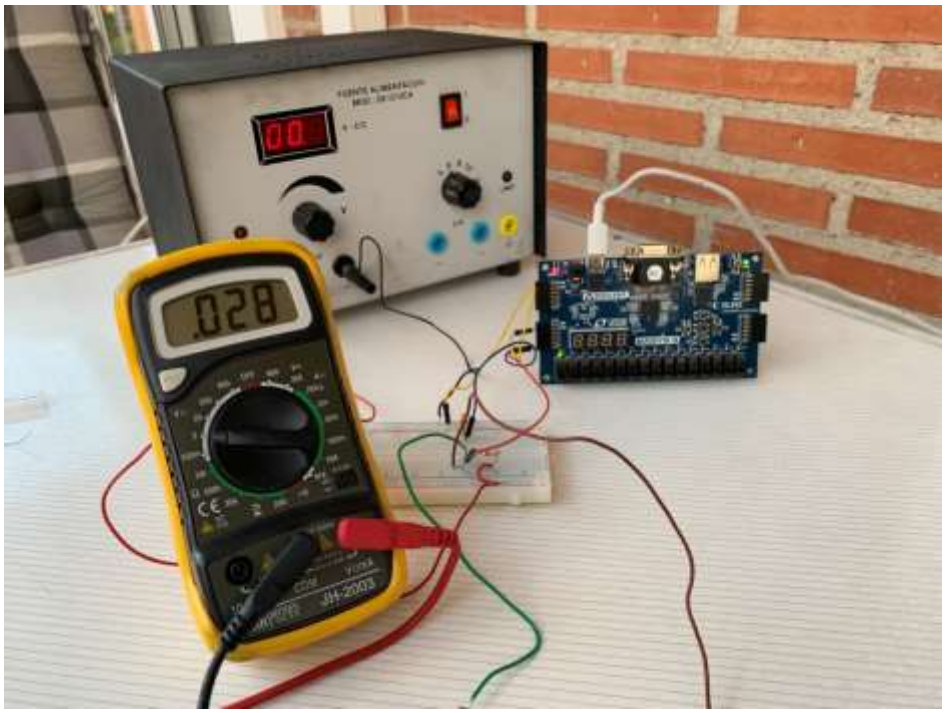


Fig. 42 Montaje para la tensión mínima

6.1.2 Prueba del bloque de la UART.

Para la realización de la prueba de comprobación del funcionamiento de la UART se realiza un ejemplo utilizando la FPGA y el ordenador. El ordenador debe contar con un terminal que permita la comunicación serie entre los dos dispositivos. Se utiliza el programa Tera Term, que es un emulador de terminal.

La UART implementada en la FPGA realiza dos funciones por un lado, transmite información de la placa al ordenador y por otro, recibe datos del ordenador en la placa.

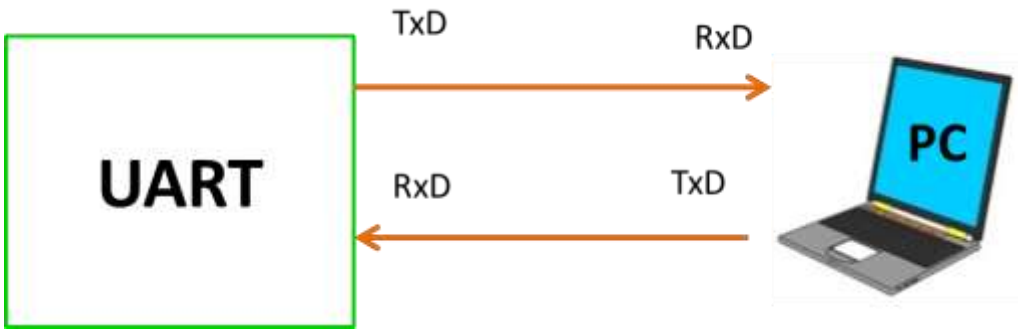


Fig. 43 Conexión UART-PC

Antes de realizar la prueba detallada a continuación con las especificaciones del sistema se realizaron pruebas sencillas para comprobar su funcionamiento. Algunas de estas pruebas fueron: transmitir continuamente un carácter desde la UART al PC y mostrarlo en el terminal; y transmitir desde el PC a la UART a través del teclado y mostrar en el terminal la secuencia transmitida.

Para realizar la demostración se envía un fichero de configuración desde el PC a la UART de la FPGA. Este fichero contiene los valores de los parámetros que son configurables en el sistema (T_{target}, T_{min}, T_{max}, Cte_1, Cte_2…).

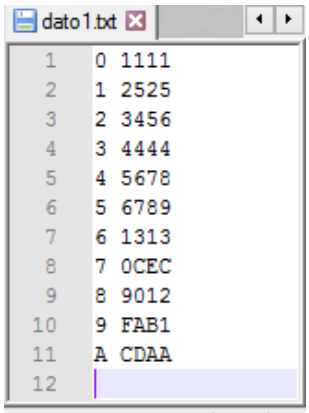


Fig. 44 Fichero de configuración

El formato del fichero es el definido para poder transmitir datos desde el PC a la FPGA. Se indica el identificador del registro a configurar 0, 1, 2…A, un espacio, y el valor del parámetro de configuración en hexadecimal, cuatro dígitos y por último un retorno de carro o fin de línea.

Una vez recibido el fichero de configuración, se envía a través del teclado del ordenador el carácter ASCII 'R' o 'r'. La FPGA recibe el carácter e interpreta su significado. En este caso, la FPGA realiza la transmisión en serie de una sucesión de caracteres entre los que se incluyen los valores de configuración recibidos previamente.

Se realiza esta comprobación repetidas veces con diferentes parámetros de configuración para ver si se reciben y se transmiten adecuadamente a través de la UART. Se prueba con valores con todos los dígitos iguales, iguales dos a dos, iguales pero alternados, dígitos aleatorios diferentes…Si el valor del parámetro de configuración en hexadecimal tiene menos de cuatro cifras, se representan con un '0' las cifras más significativas.

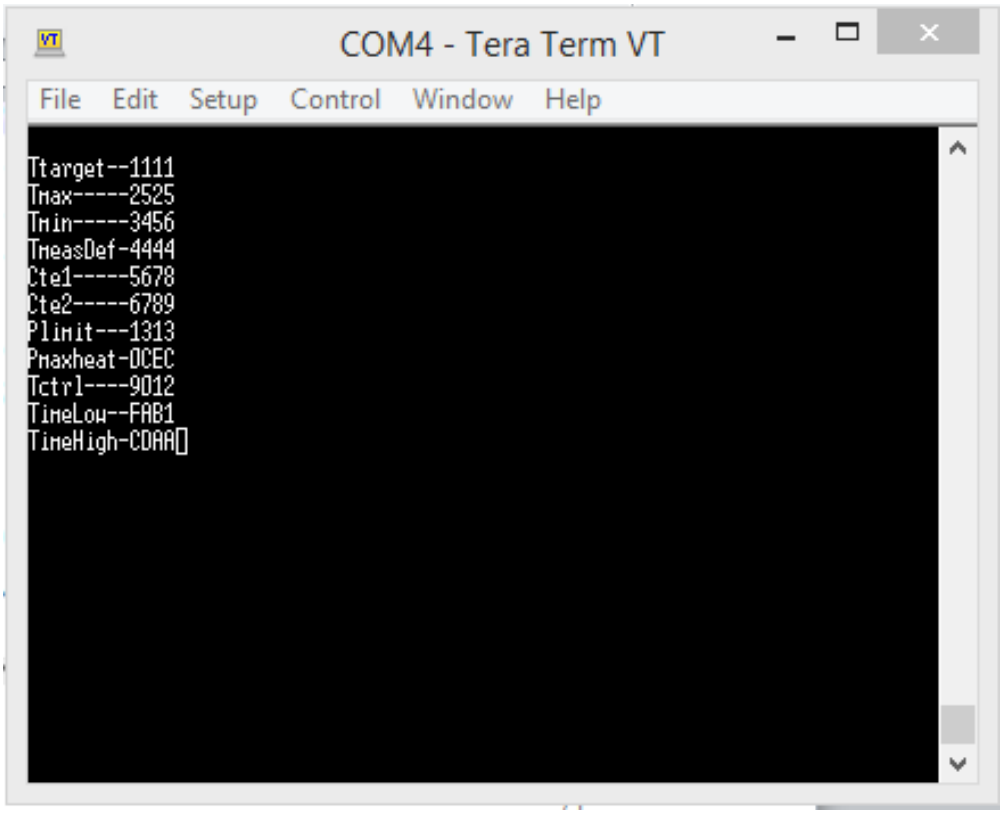


Fig. 45 Pantalla del terminal al recibir el comando 'r' o 'R'

Si se escribe en el teclado del PC una 'o' o una 'O' se muestran todos los datos almacenados. Entonces, se realiza una prueba para comprobar que se envían correctamente los datos que se guardan pero sin tener los valores correspondientes. Se realiza esta prueba antes de tener en funcionamiento el sistema completo.

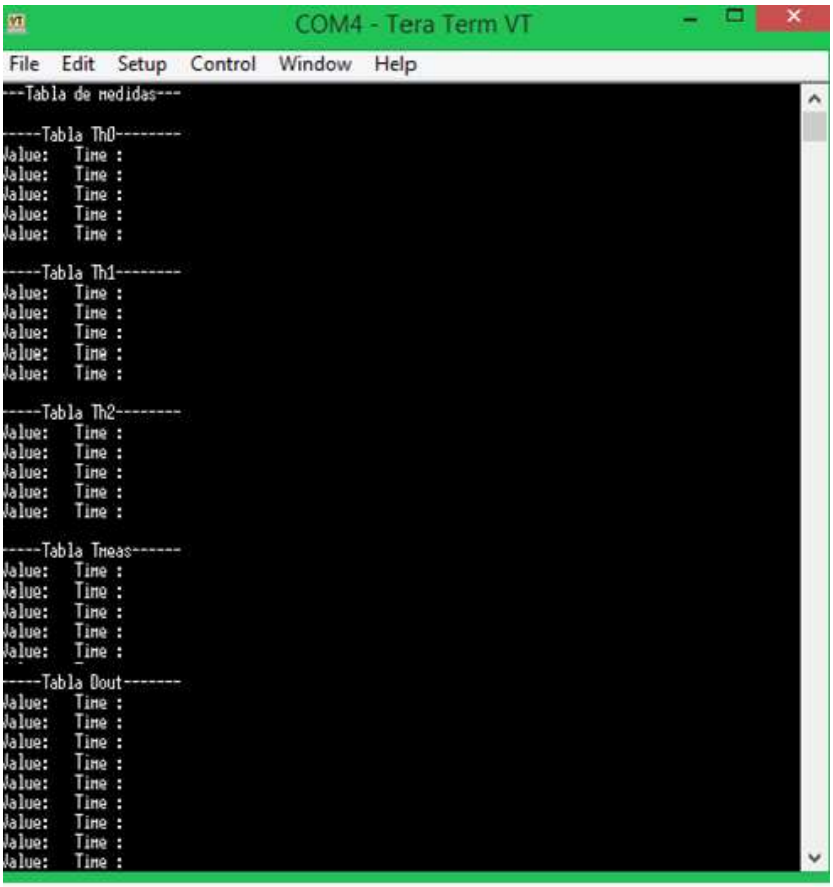


Fig. 46 Pantalla del terminal al recibir el comando ‘o’ o ‘O’

6.2 Pruebas en hojas de cálculos.

Se realizan pruebas en de hojas de cálculo para comprobar el funcionamiento del sistema en determinadas situaciones.

6.2.1 Prueba del Algoritmo de Control.

Para comprobar el funcionamiento del Algoritmo de Control se realizan cálculos en Excel. El uso de esta herramienta matemática es debido a las especificaciones recibidas por parte de la empresa SENER. Sus especificaciones de la aplicación del algoritmo estaban realizadas en Excel. Para poder comprobar el funcionamiento de lo que haría el algoritmo en VHDL, se hace previamente de forma matemática para poder comparar los resultados obtenidos con los proporcionados. [39]

Se realiza esta prueba para comprobar si la implementación digital del algoritmo de control se realiza de manera correcta, es decir, para comprobar si la resolución de bits es suficiente.

6.2.1.1 Modelo Teórico.

Originalmente, se tiene un modelo teórico de la implementación de la simulación térmica cuando se cierra el lazo.

El modelo teórico proporcionado en las especificaciones del trabajo es:

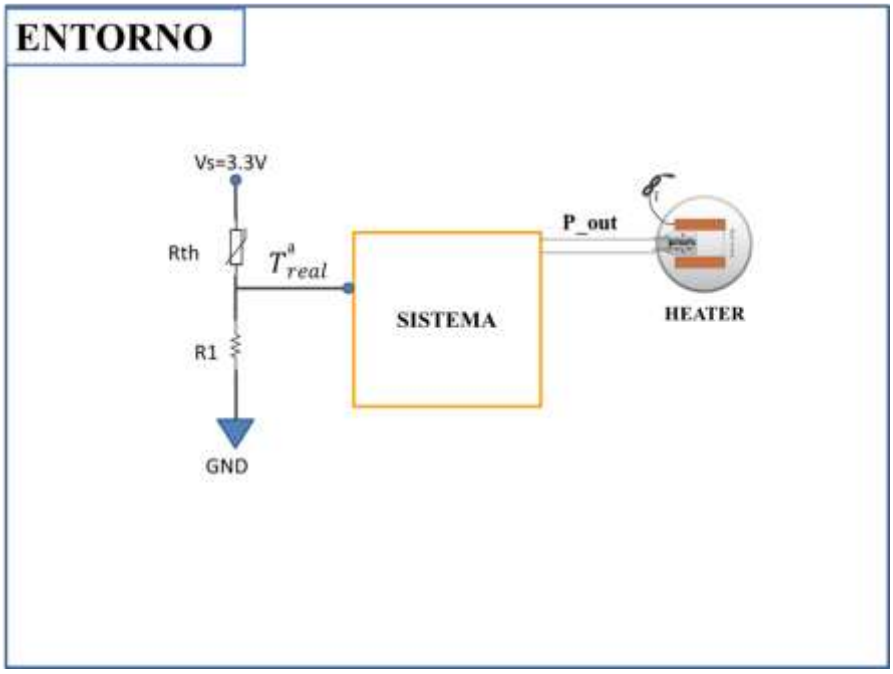


Fig. 47 Entorno teórico

Cuando se comienza a ejecutar el algoritmo se tienen unos valores iniciales de los parámetros utilizados:

* k: Número de períodos de ejecución del lazo de control. Su valor inicial es 1.
* P_out (1): Potencia comandada calculada en el primer período. Su valor es 0.

* Tmeas (1): Medida de temperatura calculada en el primer período. Su valor es Tmeas_def, la temperatura por defecto cuando se empieza a ejecutar el algoritmo.

Lo primero que se obtiene es la temperatura real, es la temperatura medida a la entrada del sistema. Esta temperatura es la temperatura medida con el circuito divisor de tensión con el termistor y la resistencia R_1.

Para calcular la temperatura real se hace a través de la siguiente fórmula:

$$T^{a}_{real}(k) = T^{a}_{heat_{sink}} - \left(T^{a}_{heat_{sink}} - \ T^{a}int(k)\right) * \ e^{\frac{0-T}{\tau 2}} \qquad (6.3)$$

donde:

$T^{a}_{real}(k)$*: Temperatura calculada en el instante k. Es* T_{meas} *(k).*

$T^{a}_{heat_{sink}}$*: Temperatura del disipador de calor (heat sink).* $T^{a}_{heat_{sink}} = 20\,℃$

$T^{a}int(k)$*: Temperatura interna en el instante k.*

T*: Período.* T=5s

$\tau 2$*: Constante térmica del material.* $\tau 2$= 100s

Para calcular la temperatura interna se hace a través de la siguiente fórmula:

$$T^{a}_{int}(k) = T^{a}_{real}(k-1) + P_{out}(k-1) * (Th_{con} * \left(1 - \ e^{\frac{0-T}{\tau}}\right) + Th_{con2} * \left(1 - \ e^{\frac{0-T}{\tau 2}}\right)) \qquad (6.4)$$

donde:

$P\_out(k-1)$: Potencia comandada calculada en el instante anterior.

$T^{a}_{real}(k-1)$: *Temperatura calculada en el instante anterior, k-1. Es* T_{meas} *(k-1).*

Th_{con}: *Conductividad térmica.* Th_{con}= 0.04219409 k/W

Th_{con2}*: Conductividad térmica.* Th_{con2}=0.02 k/W

τ*: Constante térmica del material.* τ= 0.1 s

Después se calcula la potencia calculada que se utiliza para el cálculo de la potencia de salida. La fórmula de la potencia calculada es:

$$P_{calc}(k) = P_{out}(k-1) + C_1 * \left(T_{target} - T^{a}_{real}(k)\right) + C_2 * (T_{target} - T^{a}_{real}(k-1)) \quad (6.5)$$

donde:

C_1 *: Constante 1, parámetro del lazo PI* C_1 =0.02 W/°C

C_2 *: Constante 2, parámetro del lazo PI* C_2=0.008W/°C

T_{target}: Temperatura objetivo para el lazo de control. T_{target} =22°C

La potencia de salida se calcula de la siguiente forma:

$$P_{out}(k) = \begin{cases} P_{min} & si\ P_{calc}(k) < P_{min} \\ P_{calc}(k) & si\ P_{min} < P_{calc}(k) < P_{max} \\ P_{max} & si\ P_{calc}(k) > P_{max} \end{cases} \quad (6.6)$$

donde:

P_{min} *: Potencia mínima comandable.* P_{min} = 0W

P_{max} *: Potencia máxima comandable.* P_{max}= 4 W

Por último, se calcula el ciclo de trabajo de la siguiente forma:

$$D(k) = \begin{cases} 255 & si\ P_{out}(k) > P_{\max\_heater} \\ ENTERO\left(\frac{255 * P_{out}(k)}{P_{\max\_heater}}\right) & si\ P_{out}(k) \leq P_{\max\_heater} \end{cases} \quad (6.7)$$

El ciclo de trabajo calculado con la fórmula anterior se utiliza para generar la señal de salida de la FPGA para comandar el *heater*, esta señal es cuadrada de período 100Hz y ciclo de trabajo D (k).

6.2.1.2 Implementación digital

Se realiza una simulación del algoritmo de control realizando una digitalización de los datos para comprobar cómo funciona la FPGA.

Para comprobar la implementación digital se utiliza el circuito divisor de tensión, para calcular la resistencia del termistor en todo el rango de temperatura. Se calcula utilizando la ecuación de Steinhart- Hart: [30]

$$\frac{1}{T} = \frac{1}{To} + \frac{1}{\beta} \cdot \ln\left(\frac{Rth}{Ro}\right) \rightarrow \beta \cdot \left(\frac{1}{T} - \frac{1}{To}\right) = \ln\frac{Rth}{Ro} \rightarrow Rth = Ro \cdot e^{\left(\beta \cdot \left(\frac{1}{T} - \frac{1}{To}\right)\right)} \tag{6.8}$$

donde:

Ro=10 KΩ

β=3380

To=25°C=298.15°K

T: Rango de temperatura que se varía desde -10°C a +50°C en pasos de 0.1°C

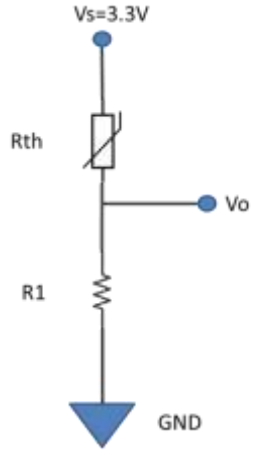


Fig. 48 Circuito divisor de tensión

Se calcula la tensión medida Vo, que se digitaliza a continuación:

$$D = Vo \cdot \frac{2^{12}-1}{Vs} \tag{6.9}$$

Al representar los valores digitales correspondientes a la tensión Vo se observa que siguen una tendencia casi lineal. Entonces, se vuelve a calcular la temperatura que se corresponde con el valor digital de tensión.

La ecuación de la recta para ver la relación entre el valor digital y la temperatura es la siguiente:

$$D = m * T + C \tag{6.10}$$

Se obtienen los valores de la pendiente (m) y la ordenada en el origen (C):

$$m = \frac{\Delta Y}{\Delta X} = \frac{Y1 - Y0}{X1 - X0} = \frac{1186 - 1148}{1 - 0} \rightarrow m = 38 \tag{6.11}$$

$$C = Y0 \rightarrow C = 1148 \tag{6.12}$$

La ecuación de la recta con los valores calculados es:

$$D = m * T + C \rightarrow$$

$$D = 38 * T + 1148 \rightarrow 38 * T = D - 1148 \rightarrow T = (D - 1071) * 0.026315789 \tag{6.13}$$

La representación gráfica de las temperaturas con sus valores digitales es:

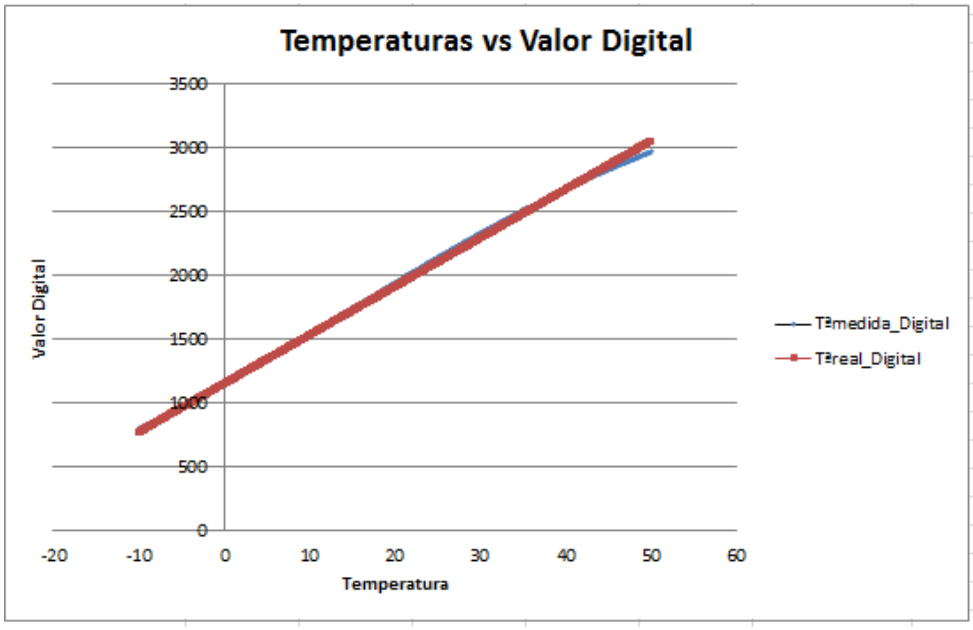


Fig. 49 Gráfico Temperatura vs Valor Digital

Se observa que hay una pequeña desviación de la temperatura digitaliza respecto a la temperatura real que se representa.

El error máximo que hay entre las temperaturas al realizarse la digitalización es:

$$Error_{max} = 2.026°C \tag{6.14}$$

El error medio es:

$$Error_{medio} = 0.6583°C \tag{6.15}$$

La desviación típica del error es:

$$Desviación_{típica} = 0.4753°C \tag{6.16}$$

Para el desarrollo de la implementación se tiene la siguiente situación:

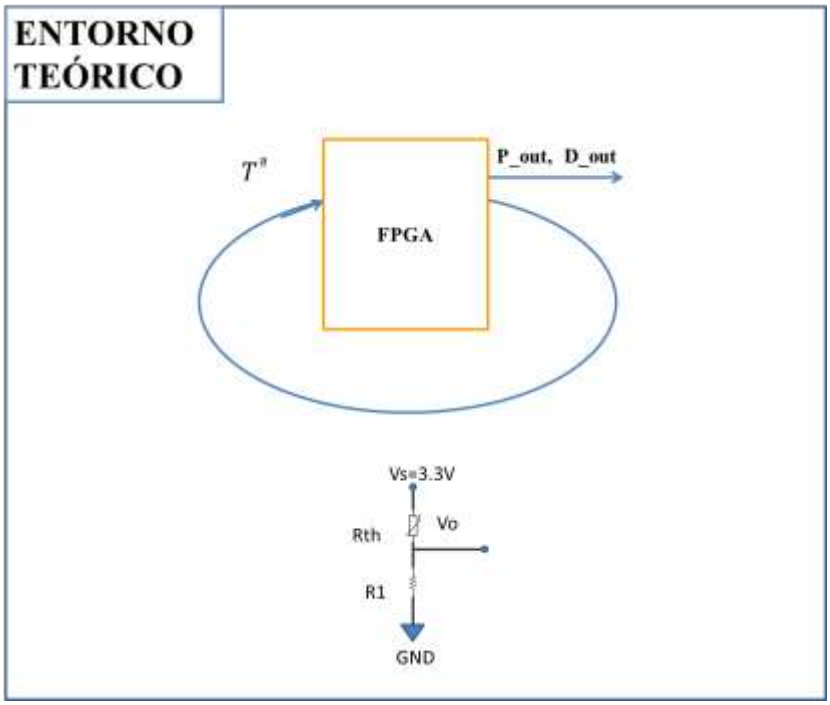


Fig. 50 Entorno simulación digital

Los pasos seguidos para comprobar el funcionamiento en simulación del algoritmo son los siguientes:

- Se partía de una temperatura real, Tmeas_def en el primera caso para comenzar la ejecución del algoritmo:

$$T^{a}_{real}(k-1) = T_{meas_{def}} = 18°\text{C} \tag{6.17}$$

- Se calcula el valor de la temperatura anterior en grados Kelvin:

$$T^{a}_{real}(k-1) = 18 + 273.15 \rightarrow T^{a}_{real}(k-1) = 291.15°\text{K} \tag{6.18}$$

- Se calcula el valor de la resistencia del termistor para la temperatura en grados Kelvin:

$$Rth = Ro \cdot e^{\left(\beta\cdot\left(\frac{1}{T}-\frac{1}{To}\right)\right)} \rightarrow Rth = 10000 \cdot e^{\left(3380\cdot\left(\frac{1}{291.15}-\frac{1}{298.15}\right)\right)}$$
$$\rightarrow Rth = 13133{,}2309\Omega \tag{6.19}$$

- Se calcula la tensión Vo para el circuito divisor de tensión, es la tensión en el instante anterior, k-1:

$$Vo = \frac{R1}{R1 + Rth} * Vs$$
$$\rightarrow Vo(k-1) = \frac{11000}{11000 + 13133{,}2309} * 3{,}3 \rightarrow Vo(k-1) = 1{,}50415V \tag{6.20}$$

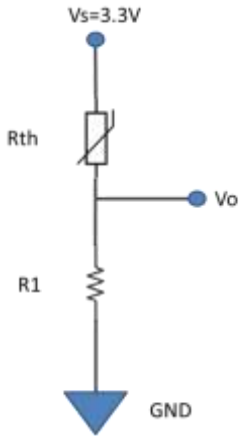


Fig. 51 Circuito divisor de tensión

❖ Se calcula el valor digital con el que se corresponde la tensión Vo, es decir el valor devuelto por el ADC:

$$D = Vo \cdot \frac{2^{12}-1}{Vs} \rightarrow D(k-1) = 1{,}50415 * \frac{2^{12}-1}{3{,}3} \rightarrow$$

$$D(k-1) = 1866.513$$

(6.21)

Se coge solo la parte entera de ese valor para obtener en valor devuelto por el XADC en el instante k-1: $XADC(k-1) = 1866$

❖ Con la aproximación de la recta realizada en el entorno teórico, se calcula la relación entre el valor digital medido con el ADC, que es un valor de tensión, y la temperatura que se mide:

$$D = m * T + C \rightarrow T = (D - C) * \left(\frac{1}{m}\right) \rightarrow$$

$$T(k-1) = (1866 - 1148) * 0{,}026315789 \rightarrow T(k-1) = 18{,}8947\,℃$$

(6.22)

Como las temperaturas son variables de 12 bits y pueden ser positivas o negativas, se necesitan 7 bits de parte entera, 1 es para el signo, y 5 bits de parte decimal. Se ajusta la temperatura para tener esa resolución.

$$T_{12bits}(k-1) = \frac{entera(18{,}8947 * 2^5)}{2^5} \rightarrow T_{12bits}(k-1) = 18{,}875\ ℃$$

(6.23)

❖ Se calcula la temperatura interna para cerrar el lazo de control a partir de la temperatura real inicial con la fórmula de las especificaciones, para el instante k :

$$T_{int}^{a}(k) =$$

$$= T^{a}_{real}(k-1) + P_{out}(k-1)$$

$$* \left(Th_{con} * \left(1 - e^{\frac{0-T}{\tau}}\right) + Th_{con2} * \left(1 - e^{\frac{0-T}{\tau 2}}\right)\right)$$

$$\rightarrow T_{int}^{a}(k) = 18 + 0 * \left(0{,}04219 * \left(1 - e^{\frac{0-5}{0{,}1}}\right) + 0{,}02 * \left(1 - e^{\frac{0-5}{100}}\right)\right) \rightarrow$$

$$T_{int}^{a}(k) = 18\,℃$$

(6.24)

❖ Se calcula la temperatura real para el instante k a partir de la temperatura interna y la fórmula correspondiente:

$$T^{a}_{real}(k) = T^{a}_{heat_{sink}} - \left(T^{a}_{heat_{sink}} - \ T^{a}int(k)\right) * \ e^{\frac{0-T}{\tau 2}} \rightarrow$$

$$T^{a}_{real}(k) = 20 - (20 - 18) * \ e^{\frac{0-5}{100}} \rightarrow T^{a}_{real}(k) = 18{,}09754115\ ℃$$

(6.25)

❖ Se realiza el mismo procedimiento a partir de esta temperatura real en el instante k, para llegar al valor de temperatura con 12 bits de resolución. Se calcula la resistencia del termistor en ese instante, la tensión medida Vo, el valor digital proporcionado por el XADC y por último el valor de temperatura con el que se corresponde el valor digital con la resolución elegida, 7 bits de parte entera y 5 bits de parte decimal.

$$T_{12bits}(k) = 19\ ℃ \quad (6.26)$$

❖ Ahora, se ejecuta la simulación del algoritmo de control, con los valores de los instantes k-1 y k, es decir, el instante anterior y el actual, de las temperaturas representadas con 12 bits. Se calcula la potencia calculada, la potencia de salida y el ciclo de trabajo de salida.

$$P_{calc}(k) = P_{out}(k-1) + C_1 * \left(T_{target} - T^{a}_{real}(k)\right) + C_2 * (T_{target} - T^{a}_{real}(k-1))$$

(6.27)

Las constantes tienen una resolución de 10 bits, 1 bit de parte entera, 8 de parte decimal y 1 bit de signo.

$$C_1 = \frac{entera(0{,}02 * 2^8)}{2^8} \rightarrow C_1 = 0{,}01953125$$

(6.28)

$$C_2 = \frac{entera(0{,}008 * 2^8)}{2^8} \rightarrow C_2 = 0{,}0078125$$

(6.29)

La potencia de salida en el primer instante de tiempo es 0W.

La temperatura $T_{target}\ es\ 22\ ℃$

La potencia calculada para el primer instante es:

$$P_{calc}(k) = 0 + 0{,}01953125 * (22 - 19) + 0{,}0078125 * (22 - 18{,}875)$$
$$\rightarrow P_{calc}(k) = 0{,}083008\ W \tag{6.30}$$

La potencia de salida se calcula de la siguiente forma:

$$P_{out}(k) = \begin{array}{ll} 0 & si\ P_{calc}(k) < 0 \\ P_{calc}(k) & si\ 0 < P_{calc}(k) < 4 \\ 4 & si\ P_{calc}(k) > 4 \end{array} \tag{6.31}$$

Como es el segundo caso la potencia de salida coincide con la potencia calculada en la primera iteración.

$$P_{out}(k) = \quad 0{,}083008W \tag{6.32}$$

El ciclo de trabajo de salida se calcula de la siguiente forma:

$$D(k) = \begin{array}{ll} 255 & si\ P_{out}(k) > 7{,}8 \\ ENTERO\left(\frac{255 * P_{out}(k)}{7{,}8}\right) & si\ P_{out}(k) \leq 7{,}8 \end{array} \tag{6.33}$$

Como es el segundo caso el ciclo de trabajo es:

$$D(k) = ENTERO\left(\frac{255 * 0{,}083008}{7{,}8}\right) \rightarrow D(k) = 2 \tag{6.34}$$

- Para las siguientes iteraciones se utiliza como temperatura real en k-1 la temperatura real del instante k de la iteración anterior, y se calcula la nueva temperatura del instante k para realizar las operaciones del algoritmo de control.

- Cuando se termina la ejecución de todas las iteraciones del algoritmo de control tras la digitalización se muestra gráficamente la temperatura real en el instante k para todas las iteraciones y la temperatura Tmeas de ese instante dada en las especificaciones, es la temperatura real del marco teórico.

  En la gráfica se observa que la temperatura una vez implementada la digitalización de los datos y la temperatura dada en las especificaciones siguen la misma tendencia con apenas variación.

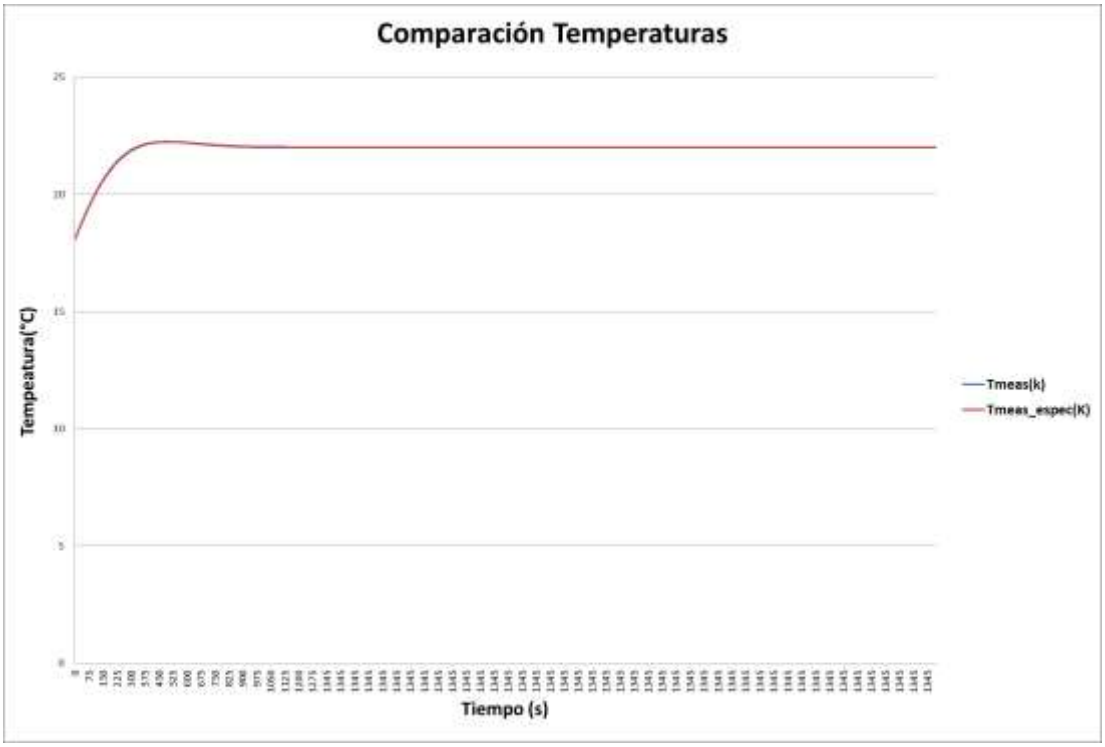


Fig. 52 Comparación de temperaturas en simulación

El error máximo en los cálculos de las temperaturas es:

$$Error_{max} = 0{,}03968°C \tag{6.35}$$

El error medio es:

$$Error_{medio} = 0.00962°C \tag{6.36}$$

La desviación típica del error es:

$$Desviación_{típica} = 0.00799306°C \tag{6.37}$$

- Los valores del ciclo de trabajo con los valores digitalizados se comprueban gráficamente tras la ejecución del algoritmo y se comparan con los valores de las especificaciones.

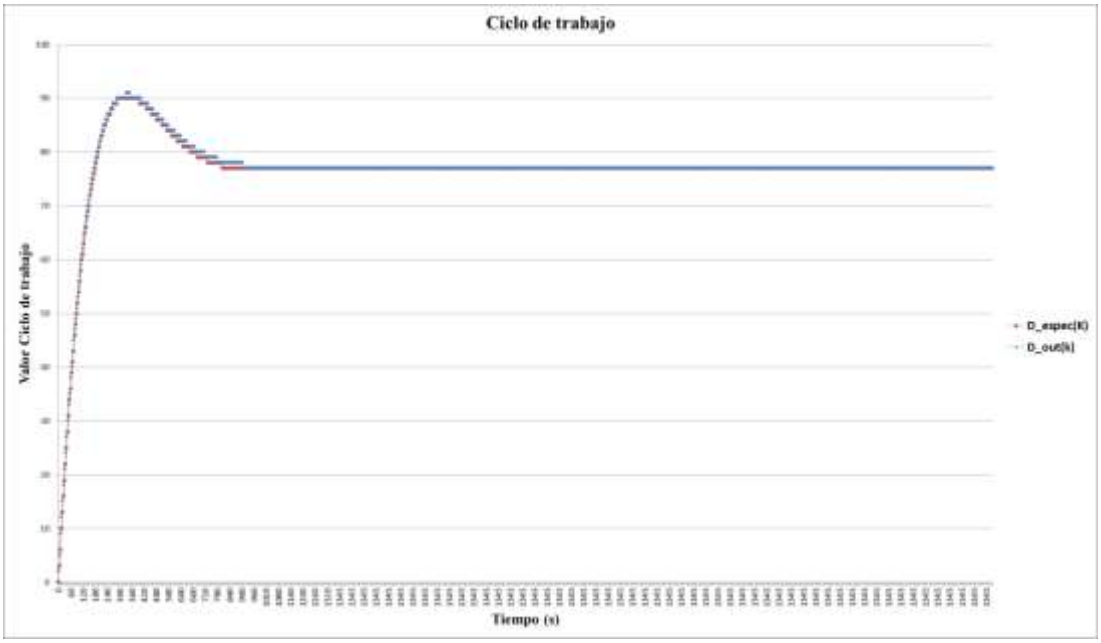


Fig. 53 Comparación del ciclo de trabajo en simulación

Se observa que hay una pequeña variación de ciclo de trabajo al digitalizar (línea azul) con respecto a las especificaciones (línea roja)

El error máximo en los cálculos del ciclo de trabajo es:

$$Error_{max} = 3 \tag{6.38}$$

El error medio es:

$$Error_{medio} = 0{,}00962072 \tag{6.39}$$

La desviación típica del error es:

$$Desviación_{típica} = 0.00799306 \tag{6.40}$$

Con los resultados obtenidos se puede concluir que con la digitalización se obtienen resultados razonablemente buenos en comparación con la simulación del algoritmo proporcionada en las especificaciones, y el error cometido tanto para el cálculo de las temperaturas como para el ciclo de trabajo es adecuado porque no es muy elevado.

6.3 Pruebas en simulación.

Se realizan pruebas con el simulador de Vivado para comprobar el correcto funcionamiento de los componentes y los bloques del sistema.

Las pruebas en simulación se realizan mediante bancos de pruebas en los que se quiere ver el funcionamiento del bloque ante unas entradas determinadas, se analizan las salidas para ver si los valores son los esperados.

6.3.1 Prueba de los Temporizadores.

Los temporizadores son un componentes que se instancian en varios de los bloques del sistema, entonces se prueba su funcionamiento por separado antes de utilizarlos en los mismos. Se instancia en el bloque de adquisición, en el de gestión de tiempo y en el algoritmo de control.

En este proyecto se utilizan dos tipos de temporizadores, por un lado un temporizador genérico para un número de bits determinado en cada instancia, y un temporizador con precarga también genérico.

El temporizador genérico es un contador ascendente hasta el número de ciclos correspondiente al número de bits que se especifica en su estancia. El temporizador con precarga es un contador ascendente que si se activa la precarga empieza a

contar desde ahí hasta el número de ciclos correspondiente al número de bits que se especifica en su estancia, en caso contrario empieza desde cero.

Se realizan varias pruebas para comprobar su funcionamiento. Con el primero de ellos, se deja que el temporizador llegue al máximo valor de la cuenta y se resetee habilitando la señal de activación del contador, se activa la señal de Clear, que resetea el contador de forma asíncrona cuando se activa, y se fijan varios valores para el máximo de la cuenta. Con el contador con precarga, se realizan las comprobaciones anteriores y una prueba adicional, la activación de señal de precarga con diferentes valores de la misma para comprobar si la cuenta empieza en ese valor.

6.3.2 Prueba del Detector de Flanco.

Esta prueba se realiza para comprobar el funcionamiento del componente que se utiliza en la gestión del tiempo.

Se realiza una simulación para comprobar que cuando se activa la señal de entrada se detecta el flanco de subida, y se activa la señal de salida.

6.3.3 Prueba de los Registros.

Se realiza esta prueba para comprobar el funcionamiento del componente *Registers*. En este caso, se fijan las entradas de datos, la habilitación de escritura y la de lectura en diferentes instantes de tiempo. Se verifica si el cambio de las señales internas que manejan las direcciones de lectura y escritura se realiza en el momento apropiado y si el valor que toman se corresponde con el teórico. Se comprueban los valores de los datos de salida, deben ser los que se han almacenado previamente.

En la simulación se puede observar cómo se van guardando los datos de entrada (data_in) en el registro indicado por la dirección de escritura (wr_addr) cada vez que se activa la señal de habilitación de la escritura. La lectura de los datos se realiza de manera combinacional, es decir, se están leyendo los datos

continuamente, pero cuando se activa la entrada de habilitación de lectura se modifica la dirección del dato que se está leyendo (rd_addr).

6.3.4 Prueba del bloque de Preprocesado.

La simulación realizada para la validación de este bloque se hace para diferentes casos en los que se verifica la votación por mayoría realizada en las temperaturas y la activación de la alarma en caso de sobrepasar los límites de temperatura.

Para la realización de esta prueba las temperaturas tienen 12 bits de los cuales se considera que 7 pertenecen a la parte entera 5 a la parte decimal, aunque se trabaja con los doce bits.

Se realizan diferentes comprobaciones:

- Comprobación 1: Todos los valores de temperatura diferentes.
  En esta comprobación se fijan tres valores de temperatura diferentes, es decir, a cada termistor se le asigna una. En la simulación se observa que el valor de T_{meas} tras la realización de la votación por mayoría de las temperaturas se corresponde con el del termistor 0 y la alarma no se activa, la temperatura está dentro del rango de temperaturas permitidas, mayores que la temperatura mínima y menores que la máxima. Previamente se ha configurado que en el caso de que no sean iguales ninguna de las tres, prevalezca la temperatura medida por el termistor 1.

- Comprobación 2: Los valores de temperatura del termistor 0 y del termistor 1 iguales.
  En esta prueba las temperaturas de dos de los termistores, el 0 y el 1, son iguales. Entonces, en ese caso la temperatura T_{meas} que se guarda en el registro se corresponde con la temperatura del termistor 0, por la configuración de este bloque. Como la temperatura T_{meas} está dentro de los límites, entonces la alarma no se activa.

- Comprobación 3: Los valores de temperatura del termistor 0 y del termistor 2 iguales.

Al realizar esta comprobación se asigna igual valor de temperatura al termistor 0 y al termistor 2, en este caso, al realizar la votación de las temperaturas se queda con la temperatura del termistor 2 que es la que se guarda en el registro T_{meas}. La temperatura almacenada no sobrepasa los rangos de temperatura fijados, entonces no se activa la alarma.

- Comprobación 4: Los valores de temperatura del termistor 1 y del termistor 2 iguales.

  En este caso, las temperaturas de los termistores 1 y 2 son iguales. Tras la realización de la votación por mayoría, la temperatura almacenada T_{meas} es la del termistor 1, como se ha definido en la votación. Como en las comprobaciones anteriores, la alarma no se activa al no estar la temperatura T_{meas} fuera de los límites de temperatura.

- Comprobación 5: Todos los valores de temperatura iguales.

  Las temperaturas de los tres termistores son iguales, en este caso, se define que la temperatura T_{meas} tome el valor del termistor 0. En esta comprobación, la temperatura T_{meas} tiene un valor dentro de los límites, entonces la alarma permanece inactiva.

- Comprobación 6: Los valores de temperatura del termistor 0 y del termistor 2 iguales y mayores que la temperatura máxima.

  En esta comprobación, las temperaturas de los termistores 0 y 2 son iguales, entonces el registro T_{meas} almacena la temperatura del termistor 2, por la votación realizada. La temperatura guardada sobrepasa el límite superior de temperatura, entonces la alarma se activa para poder detectar el fallo.

- Comprobación 7: Los valores de temperatura del termistor 0 y del termistor 1 iguales y menores que la temperatura mínima.

  En esta prueba, los termistores 0 y 1 tienen la misma temperatura, entonces el registro T_{meas} almacena la temperatura del termistor 0, por la votación realizada. La temperatura almacenada está por debajo del límite

inferior de temperatura, entonces la alarma se activa para poder detectar el fallo.

6.3.5 Prueba del bloque de Gestión de Tiempo

En esta prueba se comprueba que la gestión del tiempo se realiza correctamente, para ello se verifica el funcionamiento de tres componentes que se probaron por separado ahora todos juntos y con interconexiones entre ellos.

En esta comprobación, se comprueba que el temporizador genérico de un milisegundo está siempre contando. Cuando este contador llega al máximo que puede contar se pone a cero y vuelve a empezar, y a la vez habilita el temporizador con precarga de 64 bits para que comience a contar. Si cuando este segundo contador está contando se activa la señal de sincronismo (sync_in), que es detectada por el detector de flancos de subida, el valor de su cuenta se modifica por el de la precarga y continua contando a partir de ahí, hasta que llega al valor máximo que puede alcanzar.

En la simulación realizada no se utilizan los valores de los bits necesarios para temporizar un milisegundo, se necesitarían 17 bits, y en simulación se hace con 4 bits, para poder observar su funcionamiento es válido.

En la imagen mostrada a continuación se ve que el contador de un milisegundo está siempre contando (ena_cont1) y va de 0 a 15 porque en simulación se está usando un contador de 4 bits. Cuando se activa la entrada Sync_in_tb, el contador de 64 bits empieza a con el valor de la precarga (Time0_tb) una vez activada la señal de habilitación de este temporizador (ena_cont64).

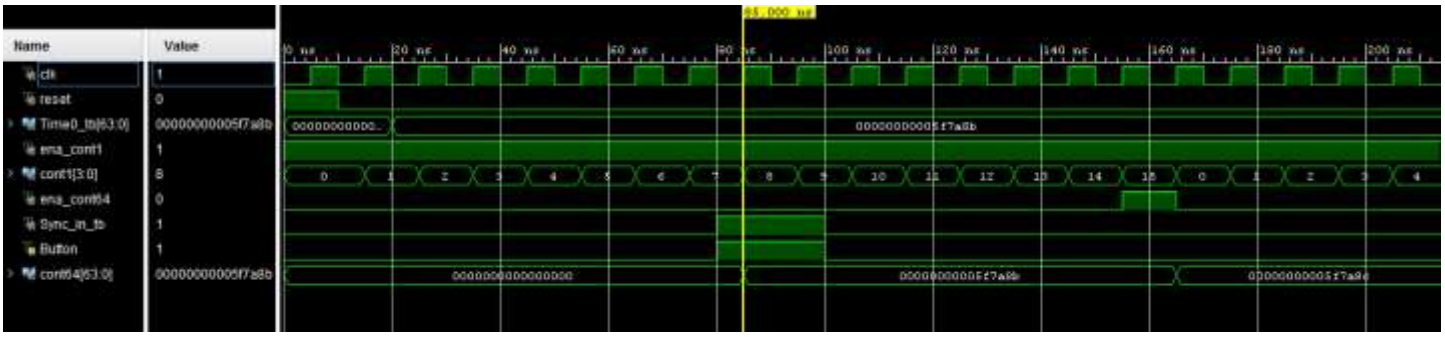


Fig. 54 Simulación del bloque de gestión de tiempo

6.3.6 Prueba del Algoritmo de Control

Se realiza una prueba en simulación para comprobar el funcionamiento del Algoritmo de Control. Esta prueba se realiza tras comprobar la simulación realizada en hojas de cálculo de lo que haría el algoritmo en la FPGA con la digitalización de los datos.

En esta prueba se siguen los mismos pasos que en la hoja de cálculo para obtener el resultado final de la potencia de salida y del ciclo de trabajo de salida para comandar el *heater*. Ahora se realiza la simulación en el lenguaje VHDL y con el simulador del software de Vivado.

Se realizan los cálculos intermedios de la ejecución del algoritmo para obtener finalmente el valor de potencia de salida y del ciclo de trabajo comparando los resultados intermedios de la potencia calculada obtenidos de cada iteración con unos valores límite de potencia fijados. Según las comparaciones realizadas se observa el valor a la salida correspondiente a esa iteración.

Además, se van registrando los valores de cada iteración de la temperatura T_{meas} y la potencia de salida para poder realizar la siguiente ejecución, ya que el algoritmo requiere de los valores de temperatura de ese instante y del anterior, y de la potencia de salida del instante anterior.

Se comprueba en cada uno de los pasos del algoritmo que los valores que se van obteniendo de cada señal se corresponden con los valores que habíamos obtenido previamente en la simulación en Excel.

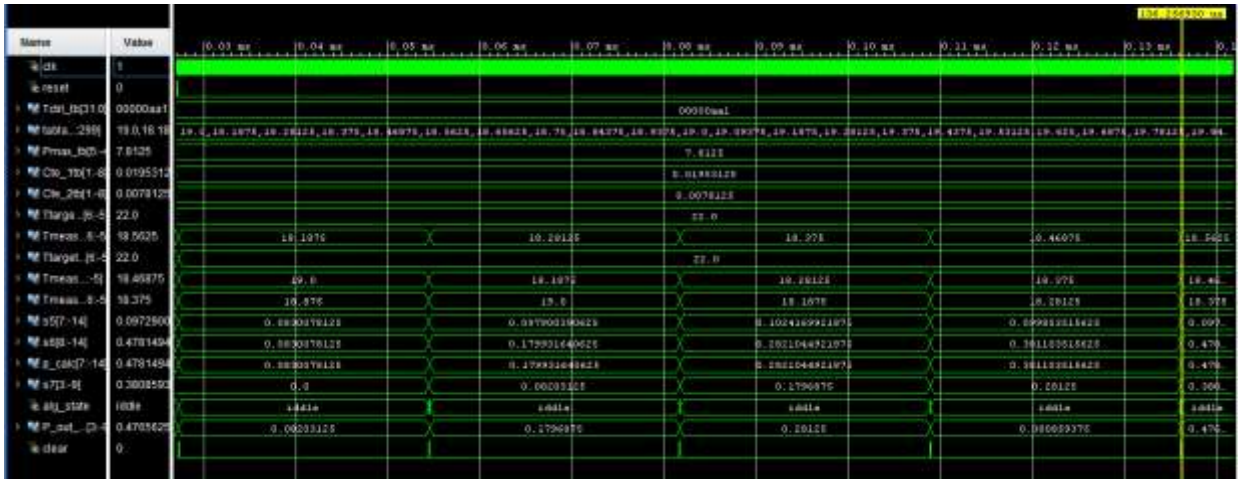


Fig. 55 Simulación del Algoritmo de Control

7 CONCLUSIONES Y TRABAJOS FUTUROS

7.1 Conclusiones

Tras la realización del trabajo de fin de grado se concluye que se han cumplido los objetivos fijados antes del comienzo de su elaboración.

Se ha conseguido hacer pruebas y comprobaciones de cada una de las partes diseñadas para la implementación de la RTU. Con la realización de estas comprobaciones se puede concluir que con la integración de todos los bloques se conseguirá el resultado deseado.

Además, se ha diseñado una solución en VHDL, que podría utilizarse para cualquier FPGA aunque en el desarrollo se haya utilizado una FPGA concreta.

Por último, como es un diseño configurable en el que todas los parámetros son modificables de manera sencilla se podría utilizar para varios desarrollos aunque tengan distintas configuraciones.

7.2 Trabajos futuros

A partir de este trabajo de fin de grado se podrían añadir nuevas líneas de trabajo para continuar en el futuro.

Se podría comprobar el funcionamiento completo del sistema realizando una comprobación experimental con los *heaters*

Además se podría añadir a otros componentes del satélite para ver su funcionamiento real integrándolo con otras partes del satélite.

También se podrían añadir más funcionalidades al proyecto realizado, como puede ser la utilización de otro tipo de sensores que permitan obtener información del estado del satélite. Se podrían añadir las siguientes funcionalidades [14]:

- La utilización de otro tipo de sensores como sensores para medir la presión.
- El control de actuadores y sensores *Attitude and Orbit Control System* (AOCS) como giroscopios, magnetómetros, GPS…
- La distribución de la potencia a los *heaters* y a las cargas activas.
- Optimización del acondicionamiento necesario para los sensores analógicos.

8 PRESUPUESTO

La estimación del coste de realización de este trabajo de fin de grado debe incluir tanto los costes de los materiales necesarios como los costes de personal.

8.1 Coste del desarrollo del proyecto.

Para analizar los costes de los materiales para el desarrollo del proyecto se tiene que valorar el coste por separado de cada uno de los componentes utilizados individualmente y el coste de las licencias de los programas utilizados.

Tabla 2 Coste de los materiales

Materiales	Unidades	Precio Unitario	Precio Total	Proveedor
FPGA Basys 3 de Xilinx	1	130,00€	130,00€	Farnell
Resistencias 10KΩ	2	0,104€	0,21€	Farnell
Polímetro Digital	1	11,95€	11,95€	Amazon
Fuente de alimentación DC	1	64,90€	64,90€	Amazon
Termistores Pt1000	3	0,60€	1,80€	Alibaba
Ordenador personal	1	450,00€	450,00€	HP
		TOTAL	658,86€	

Tabla 3 Coste de las licencias

Licencias	Precio Unitario	Precio Total
Vidado Design Suite	0,00€	0,00€
Tera Term	0,00€	0,00€
	TOTAL	0,00€

Para poder realizar el proyecto se requiere el trabajo de un ingeniero y el coste dedicado al personal del proyecto se muestra a continuación.

Tabla 4 Coste del personal

Personal	Tiempo	Precio por mes	Precio Total
Ingeniero de Telecomunicaciones	9 meses	1.800,00€	16.200,00€
		TOTAL	16.200,00€

Tabla 5 Coste total

Coste total del proyecto	Precio total
Materiales	658,86€
Licencias	0,00€
Personal	16.200,00€
TOTAL	16.858,86€

8.2 Impacto socio-económico

Se realiza el diseño e implementación de la RTU para utilizarla en satélites *low cost*. Al desarrollar este tipo de satélites se puede reducir considerablemente el coste total de un satélite, pudiendo estar al alcance de un mayor número de empresas, proyectos universitarios y agencias espaciales. Debido a la evolución de las tecnologías empleadas y al tiempo total empleado en desarrollar cada satélite se pueden lanzar un mayor número de satélites, de diferentes tipos, comunicaciones, científicos, de navegación…

Sin contar los gastos de personal es una solución que no llega a los 700€, un precio razonable y no muy elevado para formar parte de la construcción del satélite, cuyo diseño, construcción y lanzamiento puede llegar a ser de 500 millones de euros. [6]

Por tanto, esta implementación de bajo coste sería una buena alternativa en los nuevos satélites que se están desarrollando.

9 BIBLIOGRAFÍA

[1] A. Llorente, "¿Cuántos satélites hay orbitando la Tierra y cómo es posible que no choquen?", 2018. [En línea]. Disponible en: https://www.bbc.com/mundo/noticias-46408633

[2] (UNOOSA) United Nations Office for Outer Space Affairs, "Online Index of Objects Launched into Outer Space.".[En línea]. Disponible en: http://www.unoosa.org/oosa/osoindex/search-ng.jspx?lf_id=#?c=%7B%22filters%22:%5B%5D,%22sortings%22:%5B%7B%22fieldName%22:%22object.launch.dateOfLaunch_s1%22,%22dir%22:%22desc%22%7D%5D,%22match%22:null%7D

[3] Esther, "Tipos de Satélites Artificiales.", 2010. [En línea]. Disponible en: http://satlitesartificiales.blogspot.com/2010/10/tipos-de-satelites-artificiales.html

[4] "Satélites artificiales: nombres, tipos, características y mucho más. ", 2018. [En línea].Disponible en: http://misistemasolar.com/satelites-artificiales/

[5] J. C. Cortés, "España en el futuro 'New Space' y 'Space Industry 4.0'", 2019. [En línea].Disponible en: http://www.revistanoticias.sener/news/espana-en-el-futuro-new-space-y-space-industry-4-0-juan-carlos-cortes/54/

[6] "A Basic Guide to Nanosatellites". [En línea]. Disponible en: https://alen.space/basic-guide-nanosatellites/. [Acceso: 30-Mayo-2019]

[7] "What are SmallSats and CubeSats? ", NASA, 2015 [En línea].Disponible en: https://www.nasa.gov/content/what-are-smallsats-and-cubesats

[8] R. Taylor, "Satélites pequeños y baratos para saber qué hacemos cada día", 2014. [En línea].Disponible en: https://www.bbc.com/mundo/noticias/2014/05/140516_tecnologia_satelites_caja_zapatos_mz

[9] P. Morillo, "Descubre el interesante mundo de los satélites", 2017. [En línea].Disponible en: https://futurizable.com/satelites/

[10] J. M. Alonso San Sebastián, "La idea española para lanzar mini-satélites y reducir la contaminación de los coches", 2019. [En línea].Disponible en: https://www.elconfidencial.com/tecnologia/2019-03-23/lanzador-europeo-satelites-tecnalia-pld-space-proyecto-smile_1899226/

[11] B. “A satellite engineer explains the challenges of space electronics.”, 2018 [En línea].Disponible en: http://blog.snapeda.com/2018/08/16/engineer-spotlight-brady-salz-from-astranis/

[12] L. A. Silva Rubio, A. Bermúdez Huertas, P. Mojica G., S. Cuéllar, y C. Medina, “Boletín tecnológico Nanosatélites”, 2017 [En línea].Disponible en: http://www.sic.gov.co/sites/default/files/files/Propiedad%20Industrial/Boletines_Tecnologicos/Boletin_nanosatelites_29junio.pdf

[13] I. Melgar Bautista, "Documentos internos de la Cátedra UC3M-SENER", 2019.

[14] ESA, "Remote Terminal Units", 2019. [En línea].Disponible en: https://www.esa.int/Our_Activities/Space_Engineering_Technology/Onboard_Computer_and_Data_Handling/Remote_Terminal_Units. [Acceso:28-Mayo- 2019]

[15] ESA, "Architectures of Onboard Data Systems», European Space Agency",2019. [En línea]. Disponible en: http://www.esa.int/Our_Activities/Space_Engineering_Technology/Onboard_Computer_and_Data_Handling/Architectures_of_Onboard_Data_Systems. [Accedido: 10-jun-2019].

[16] “Digital Sensor Bus,” 2019. [En línea].Disponible en: https://www.esa.int/Our_Activities/Space_Engineering_Technology/Onboard_Computer_and_Data_Handling/Digital_Sensor_Bus/(print). [Accedido:15- mayo- 2019]

[17] K. Medrano, D. Altuve, K. Belloso, y C. Bran, «Development of SCADA using a RTU based on IoT controller», en IEEE ICA-ACCA 2018 - IEEE International Conference on Automation/23rd Congress of the Chilean Association of Automatic Control: Towards an Industry 4.0 - Proceedings, 2019. [En linea]. Disponible: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85062188134&doi=10.1109%2fICA-ACCA.2018.8609700&partnerID=40&md5=01b9e0d18eb2733d1af03a22551fd75d. [Accedido: 15-Mayo-2019]

[18] C. Tato Ribera, "Documento de especificaciones del Control Térmico," 2019.

[19] S. Phan Lung, “Microprocesadores vs Microcontroladores vs FPGA.” 2015 [En línea]. Disponible en: http://idielectronica.blogspot.com/2015/04/microprocesador-vs-microcontrolador-vs.html

[20] GENERA Tecnologías, "FPGA vs otros Chips o Dispositivos Electrónicos". [En línea]. Disponible en: https://www.generatecnologias.es/fpga_vs_mcu.html. [Accedido: 08-jun-2019].

[21] E. Rojas," ¿Qué son los fpgas? ¿Cómo funcionan? ¿Para qué sirven? ¿Quién debería utilizarlos?", 2016. [En línea].Disponible en: https://nodoelectronico.com/2016/01/01/que-son-los-fpgas-como-funcionan-para-que-sirven-quien-deberia-utilizarlos/

[22] "IEEE Draft Standard for VHDL Language Reference Manual." pp. 1–824, 2018.

[23] "P1076/D6, Jun 2018 - IEEE Draft Standard for VHDL Language Reference Manual - IEEE Standard". [En línea]. Disponible en: https://ieeexplore-ieee-org.biblioteca5.uc3m.es/document/8410188. [Accedido: 15-jun-2019].

[24] S. Hobe, "The Impact of New Developments on International Space Law (New Actors, Commercialisation, Privatisation, Increase in the Number of "Space-faring Nations")", vol. 15, n.º 3-4, pp. 869-881, 2010. [En línea].Disponible en: http://www.unoosa.org/pdf/pres/2010/SLW2010/02-12.pdf

[25] Comisión Europea, "Política de la UE sobre industria espacial. Aprovechar el potencial de crecimiento económico en el sector espacial.",2013. En línea].Disponible en: https://eur-lex.europa.eu/legal-content/ES/TXT/PDF/?uri=CELEX:52013DC0108&from=EN

[26] "European Cooperation for Space Standardization". [En línea]. Disponible en: https://ecss.nl/. [Accedido: 15-jun-2019].

[27] Media ATN, "Normativa de EMC en Aeronáutica y Espacio",Alter Technology,2018. En línea].Disponible en: https://wpo-altertechnology.com/es/normativa-de-emc-en-aeronautica-y-espacio/

[28] SRC, "Tipos de Sensores de Temperatura." [En línea].Disponible en: https://srcsl.com/tipos-sensores-temperatura/

[29] E. J. A. Rodríguez, J. W. M. Ocampo, y C. A. S. Ortega, "Medición de temperatura: sensores termoeléctricos," vol. 1, no. 34, 2007.

[30] P. Kane, "Thermistors/Temperature Measurement with NTC Thermistors" [En línea]. Disponible en: https://www.jameco.com/Jameco/workshop/TechTip/temperature-measurement-ntc-thermistors.html

[31] SatPro.tv, "Satellite Dish Heaters Snow/Ice", SatPro.tv. [En línea]. Disponible en: http://www.satpro.tv/.

[32] NASA, "07. Thermal Control – State of the Art of Small Spacecraft Technology", 2019. [En línea]. Disponible en: https://sst-soa.arc.nasa.gov/07-thermal

[33] European Space Agency (ESA), "Thermal Control". [En línea]. Disponible en: https://www.esa.int/Our_Activities/Space_Engineering_Technology/Thermal_Control. [Accedido: 07-jun-2019].

[34] E. Escobar, M. Diaz, y J. C. Zagal, "Evolutionary design of a satellite thermal control system: Real experiments for a CubeSat mission", vol. 105, pp. 490-500, 2016. [En línea]. Disponible en: https://linkinghub.elsevier.com/retrieve/pii/S1359431116303167

[35] M. J. Rivas Martínez, "Gestión térmica de sistemas espaciales". [En línea].Disponible en: http://www.madrid.org/bvirtual/BVCM001846.pdf

[36] A. Karlsson, "XMM's Data-Handling Subsystem," no. 100, 1999 [En línea].Disponible en: https://www.esa.int/esapub/bulletin/bullet100/KARLSSON.pdf

[37] H. J. Kramer, "SMART-1 - eoPortal Directory - Satellite Missions». [En línea]. Disponible en: https://earth.esa.int/web/eoportal/satellite-missions/s/smart-1. [Accedido: 09-jun-2019].

[38] D. Guzman et al., "High Reliable Remote Terminal Unit for Space Applications, en 2009 12th Euromicro Conference on Digital System Design, Architectures, Methods and Tools", 2009, pp. 488-493.

[39] C. Tato Ribera,"Algoritmo de control especificaciones", 2019.

[40] GMA, "Basys "3, 2014.

[41] "Basys 3™ FPGA Board Reference Manual." Diligent, 2016[En línea]. Disponible en: https://reference.digilentinc.com/reference/programmable-logic/basys-3/reference-manual

[42] "7 Series FPGAs and Zynq-7000 SoC XADC Dual 12-Bit 1 MSPS Analog-to-Digital Converter." Xilinx, Estados Unidos, 2018 [En línea]. Disponible en: https://www.xilinx.com/support/documentation/user_guides/ug480_7Series_XADC.pdf

[43] "Vivado Design Suite User Guide Logic Simulation." Xilinx, 2018 [En línea].Disponible en: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2018_1/ug900-vivado-logic-simulation.pdf

[44] "Vivado Design Suite User Guide: Synthesis." Xilinx, 2018 [En línea]. Disponible en: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2018_1/ug901-vivado-synthesis.pdf

[45] J. M. Mendías Cuadros, "Tema 5: Especificación usando VHDL'08 Diseño automático de sistemas,"2016. [En línea]. Disponible en: http://www.fdi.ucm.es/profesor/mendias/DAS/docs/DAStema4.pdf

[46] T. León Castillo, "Conversión analógica digital y digital analógica". [En línea].Disponible en: https://www.academia.edu/27170429/CONVERSI%C3%93N_ANAL%C3%93GICA_DIGITAL_Y_DIGITAL_ANAL%C3%93GICA

[47] D. Maye, "Components". 2016. [En línea]. Disponible en: https://www.youtube.com/watch?v=nGmj1MuYB5s

[48] D. Maye, "Basys 3 xADC explanation". 2016 [En línea].Disponible en: https://www.youtube.com/watch?v=bWWxZ9fWnKg

[49] D. Maye, "Entity Description.", 2016. [En línea].Disponible en: https://www.youtube.com/watch?v=3mqEiOb2X24

[50] J. Lewis and D. Bishop, "Fixed and Floating Point Packages." 2005 [En línea].Disponible en: http://klabs.org/mapld05/presento/189_lewis_p.pdf

[51] J. Jankovic, M. Subotic, and V. Marinkovic, "One solution of the accurate summation using fixed-point accumulator." pp. 508–511, 2015.

[52] G. B. Wakhle, I. Aggarwal, and S. Gaba, "Synthesis and Implementation of UART Using VHDL Codes," presented at the 2012 International Symposium on Computer, Consumer and Control, 2012, pp. 1–3 [En línea].Disponible en: https://ieeexplore.ieee.org/document/6228233

[53] M. Poorani and R. Kurunjimalar, "Design implementation of UART and SPI in single FGPA," presented at the 2016 10th International Conference on Intelligent Systems and Control (ISCO), 2016, pp. 1–5.

[54] Y. Fang and X. Chen, "Design and Simulation of UART Serial Communication Module Based on VHDL." pp. 1–4, 2011.

[55] S. Dharmendra, S. Upendra, S. Karan, K. Yash, L. Shubham, and N. Aayush, "Verilog based uart system design," vol. 6, no. 2, p. 34, 2018 [En línea]. Disponible en: https://search.proquest.com/docview/2167292347

[56] M. W. Brown, J. A. Leendertz, A. P. Makepeace, and B. J. McCabe, "Uses of a UART (universal asynchronous receiver/transmitter) integrated circuit for inexpensively coding large amounts of information on one f.m. tape channel [proceedings," vol. 277, pp. 40P-41P, 1978.

ANEXOS

Summery

Introduction

The artificial satellites are increasingly numerous in space and they have diverse missions and functions. They can be scientific satellites, communications or military among others.

Nowadays, artificial satellites of smaller size, weight, shorter period of development time are being built, for this reason the cost is lower than the ancient satellites. This kind of satellites is known as nanosatellites. For all these characteristics, more companies and universities can access to projects that involve the development and launch of this type of satellites.

The work is being developed within the framework of the investigation Chair of the University Carlos III of Madrid with the Spanish company SENER, and the final objective is to design and to build a low-cost satellite in less time than traditional satellites.

The cost of the project development is approximately 17,000€.

The main objective of this project is the design and development of a Remote Terminal Unit (RTU) in Field Programmable Gate Array (FPGA). This device was chosen for the possibility to parallelize the tasks.

State of the art

The RTU is a unit present in spacecraft and satellites of medium and large size. It is mainly responsible for interconnecting sensors, actuators and digital interfaces with the On Board Computer (OBC). In addition, it controls the functioning of the actuators and the acquisition of analogue data, in other words, it is a distributed control system in the satellite. Finally, it is a unit connected to the OBC by serial communication and at the same time it is controlled by it. The RTU must carry out the thermal control of the satellite.

The thermal control of a satellite is an essential function. The temperature of the satellite must have a certain range for the constituent components (electronic

equipment, payload…) to work properly and optimally. The thermal control system is responsible for maintaining an adequate temperature inside the satellite.

The RTU has external components to carry out the thermal control function, such as heaters and thermal sensors.

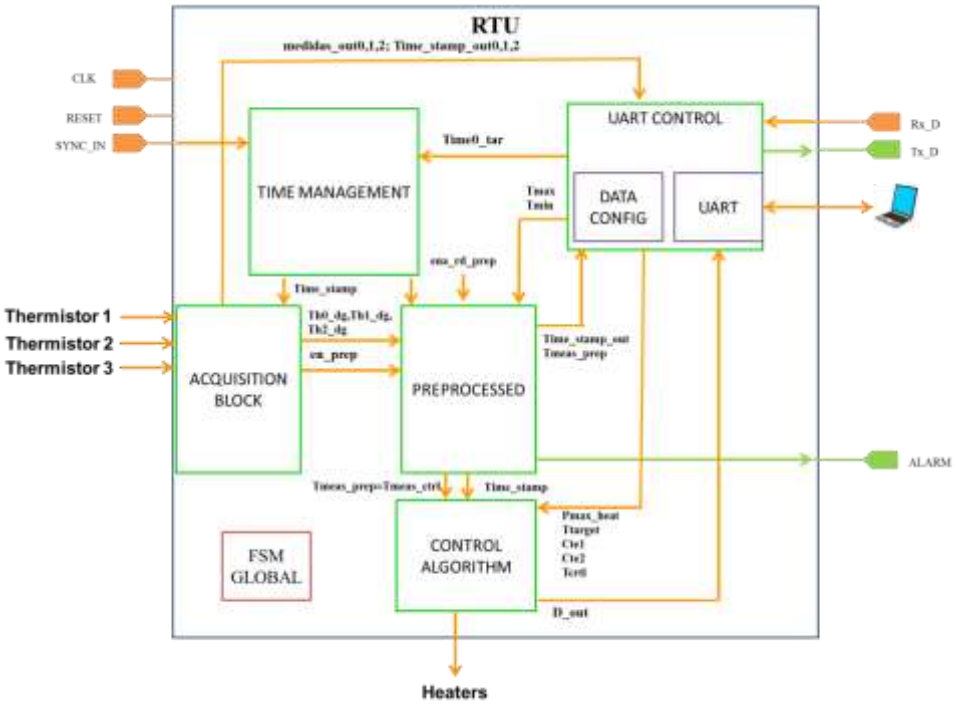
Heaters maintain the temperature of the satellites batteries or payloads at appropriate values in the cold periods of the orbit, for instance when there is an eclipse.

Thermal sensors are devices with the ability to detect changes in the temperature of the environment; these variations are transformed into electrical signals that are processed by an electronic device. One type of the temperature sensors is the resistive sensor; this kind is formed by the Resistance Temperature Detector (RTD) and the thermistors. The main advantage of the RTD is the linearity of the voltage against the temperature in a wide range of them. Pt1000 sensors will be used in the project; they have a high sensitivity, accuracy and reliability. In the event that the implementation of the project is carried out, the standards and normalizations of the European Space Agency (ESA) as defined by the European Cooperation for Space Standardization (ECSS) will be applied.

Specifications and design

The design of the RTU is made with certain characteristics provided by SENER to achieve certain technical specifications. It is made on a Basys 3 FPGA of Xilinx, and it is used the hardware description language VHDL.

The design has several blocks related to each other, each block has a set of inputs and outputs, and all of them have the input of the global clock (Clk) and the initialize of the system (Reset).



The RTU block is responsible for connecting all the blocks that are part of the system. Internal signals are used to connect all the components; these signals connect the outputs of some blocks with the inputs of others.

In this block, a state machine is used to control the system globally. Moreover, it is responsible for correct functioning of all the components that are instantiated in the RTU block.

The state machine of this block is designed to control the component of the registers that store data that is the Registers component of the acquisition block and the preprocessing block.

The temperature acquisition block is responsible for acquiring the analogue data of the temperatures of three external thermistors. These data are acquired consecutively until obtaining the values of the three channels that form the sequence.

This block has several instanced components: the XADC, the Registers and the Timer. The XADC is the analogue digital converter (ADC) of the FPGA, whose configuration to work is chosen to convert three channels sequentially that are selected with the internal multiplexer, until a conversion does not finish does not start the next one. The Registers component is used to store the acquired data of the

thermistors by the ADC. The timer is responsible for counting the number of cycles equivalent to a second of time which is the time that must pass between two conversions.

In addition, it has two state machines to control conversions. One of them, controls the conversion of the sequence of channels, it is carried out every second. The second one controls the conversion of each ADC channel.

The time management block is responsible for the synchronization of the whole system. In this block, an exit time label is configured; this label depends on two internal timers. The first one is incremented every millisecond and at the same time it modifies the timer in charge of the time label. The second one is preloaded with a certain value if the Sync_in system input is activated; also an edge detector is used to avoid possible rebounds in this input.

The Universal Asynchronous Receiver and Transmitter (UART) is chosen as the communication interface. It is used to configure the thermal control function. Through the interface you can access to the configuration parameters and the stored data. In addition, it allows loading and downloading the data of the mathematical operations of the function.

The communication block is the component of the UART, which is responsible for the transmission and reception of the interface and has the necessary configuration. In addition, it has two state machines, one to perform the transmission of data from the FPGA to the PC, it is used to display in a terminal the configuration parameters of the system and the data stored in the registers; and another for the reception, the PC communicates with the FPGA, the data is transmitted and they are processed on the board. Through the FPGA keyboard you can receive several commands that perform one function or another when analyzed. Some of the allowed commands are 'r', 'o', 'R', 'O', '1' or '2'.

The block of the control algorithm is executed to calculate the power delivered to the heater and to calculate the duty cycle (Duty Cycle) necessary to command it.

In this block the operation of the closed loop algorithm is simulated taking into account the heaters. A series of mathematical operations are performed with the temperature values that are calculated, two thermal constants and with the target temperature set. Also, some power limits are taken into account when the algorithm

is executed. After these operations, the power value is obtained to command the heater that is later used in the calculation of the dutty cycle.

This block has the timer component to start executing the algorithm, and a state machine to record the values of the data needed for the algorithm calculations such as the temperatures that are calculated in each iteration, and the power delivered to the heaters. The output power and duty cycle are calculated by making comparisons with power limit values.

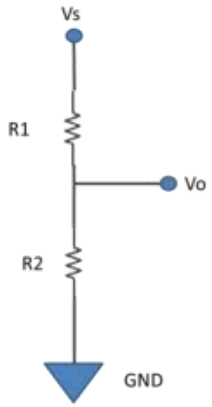
The preprocessing block is responsible for determining the average temperature of the thermistors, Tmeas. To define the average temperature is defined by a majority voting system. The defined voting system is based on three equality comparators; the thermistor 0 has priority over the rest, then in case of all the different values, that temperature will prevail.

Validation and experimental tests

Tests are made to validate the design; the correct functioning of the system and of the blocks that are part of the complete system is checked.

Tests of different types are done: tests in simulation, tests in the FPGA and tests in spreadsheets. The simulation tests are made in a software simulation tool, Vivado® Simulator. It uses the VHDL language. The tests with the FPGA need external hardware assemblies in some cases.

The tests with the FPGA are: the XADC test and the UART block test. The first one is done to verify the correct functioning of the digital analog converter, of the XADC module. The conversion of a single channel of the XADC is performed. The voltage of the source varies from 0V to a maximum of 3.3V, which is the maximum voltage that can be measured with the XADC. The voltage Vo is measured with a multimeter and the digital value corresponding to the measured voltage value is calculated. Next, it is verified that the digital value calculated numerically corresponds to the value returned by the ADC. The output of the ADC is shown by the LEDs of the FPGA. After testing several voltage values we can verify that the digital analog converter module works correctly.



A check of the UART is made with the FPGA and the computer, which must have a terminal emulator installed to allow serial communication between both devices. Some of these tests were: continuously transmit a character from the UART to the PC and display it in the terminal; and transmit the configuration file from the PC to the UART via the keyboard and show the transmitted sequence, the configured parameters and the stored data in the terminal. The correct operation of all possible commands is verified in both cases, to transmit from the FPGA to the PC and from the PC to the FPGA.

The simulation tests check the operation of each of the components that are part of the system.

Several tests of the timers check their functioning. The timer is allowed to reach the maximum value of the account by enabling the counter activation signal Enable, the Clear signal is activated, which resets the counter asynchronously when it is activated, and several values are set for the maximum of the timer. The preloaded timer makes the same checks that the previous timer and an additional test, the activation of preload signal with different values and in different moments to check if the account starts at these values.

The simulation of the edge detector checks when the input signal is activated, the rising edge is detected and the output signal is activated.

The simulation test of the Registers component is made to check this component. In this case, the data entries, the write enable and the read enable are set at different times. It is verified if the change of the internal signals that handle the read and write addresses is done at the appropriate time and if the value they take

corresponds to the theoretical one. The values of the output data are checked, they must be the ones that have been previously stored.

A test is carried out for the validation of the preprocessing, different cases are verified in which the vote is verified by a majority made in the temperatures and the activation of the alarm in case of exceeding the temperature limits.

For the time management block, a simulation is made with the combination of three previously checked components, the two kind of timers and the edge detector.

The control algorithm is checked by speadsheets tests. In order to verify the operation of what the algorithm would do in VHDL, we previously performed it mathematically in order to compare the results obtained with those provided. This test checks if the digital implementation of the control algorithm is done correctly, in other words to check if the bit resolution is enough. It is based on a theoretical model that is later digitized to see if the same results are obtained.

Conclusion and future works

It has been achieved to make tests and checks of each of the parts designed for the implementation of the RTU. It can be concluded that with the execution of these checks all the blocks will achieve the desired result.

The design made in VHDL could be implemented in any FPGA and for different parameters because it is a configurable design.

After completing the degree project, it is concluded that the objectives set have been achieve: design and development of a low-cost solution.

With the development of this type of satellites, the total cost of a satellite can be reduced considerably, being able to be available to a greater number of companies, university projects and space agencies. Due to the evolution of the technologies used and the total time spent in developing each satellite can be launched a greater number of satellites, of different types, communications, scientific, navigation…The cost of materials used would not reach € 700, a reasonable price and not too high to be part of the construction of the satellite, whose design, construction and launch can reach 500 million euros. Therefore, this low cost

implementation would be a good alternative in the new satellites that are being developed.

Finally, it is highlighted that from this work new lines of work are opened to optimize or add new functionalities, for instance: the use of other types of sensors or the checking of the operation of the complete system with the heaters.